НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Інститут телекомунікаційних систем

Кафедра Телекомунікаційних систем

«До захисту допущено»
Завідувач кафедри
_____ Леонід УРИВСЬКИЙ
«___»_____20__ р.

Дипломна робота

на здобуття ступеня бакалавра

зі спеціальності 172 Телекомунікації та радіотехніка

на тему: «Аналіз та побудова ресурсоефективних рішень IoT»

Виконав:
студент IV курсу, групи ТС-61
Мальцев Андрій Георгійович _____

Керівник:
Доц. кафедри ТС
К.т.н., доц.
Мошинська Аліна Валентинівна _____

Рецензент:
Проф. кафедри ТК
Д.т.н., проф.
Лисенко Олександр Іванович _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Інститут телекомунікаційних систем

Кафедра Телекомунікаційних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 172 Телекомунікації та радіотехніка

Програма професійного спрямування (спеціалізація) – «Телекомунікаційні системи та мережі»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Леонід УРИВСЬКИЙ

«___»_____20__ р.

ЗАВДАННЯ

на дипломну роботу студенту

Мальцеву Андрію Георгійовичу

1. Тема роботи «Аналіз та побудова ресурсоефективних рішень IoT», керівник роботи Мошинська Аліна Валентинівна, к.т.н., доцент, затверджені наказом по університету від 30 березня 2020 р. № 924-с

2. Термін подання студентом роботи 12 червня 2020 р.

3. Вихідні дані до роботи: технічні засоби всього стеку OSI для дизайну, розробки та реалізації комплексних завершених рішень IoT

4. Зміст роботи:

1) Обгрунтування актуальності теми дослідження.

2) Побулова ресурсоефективного IoT – рішення на прикладі плати Elecrow ESP8266

3) Модернізація ресурсоефективного рішення

4) Огляд нових датчиків.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо):

1 презентація, містить 15 слайдів

6. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Обґрунтування актуальності теми дослідження. Аналітичний огляд інформаційних матеріалів.	1 тиждень	Виконано
2	Побудова ресурсоефективного IoT рішення. Огляд, послідовність створення.	2 тижні	Виконано
3	Модернізація ресурсоефективного IoT рішення. Передача показників на сервер (Blynk). Передача показників на сайт кафедри	3 тижні	Виконано
4	Тест і програмування нових датчиків ресурсоефективного IoT рішення: - Огляд нових датчиків - Програмування - Тести датчиків	4 тижні	Виконано
5	Побудова більшої кількості пристроїв ресурсоефективного IoT рішення та їх синхронізація	5тижнів	Виконано

Студент Андрій МАЛЬЦЕВ

Керівник роботи Аліна МОШИНСЬКА

РЕФЕРАТ

Текстова частина дипломної роботи: 90 с., 26 рис., 1 табл., 15 джерел

Мета роботи - створення ресурсоефективного IoT-рішення для отримання показників вологості та температури повітря аудиторій у тридцятому корпусі Київського Політехнічного Інституту.

В даній роботі розглядається IoT як технологія майбутнього, його перспективи та недоліки.

Оглянуто найпростіші IoT – рішення, покроково описано алгоритм створення і програмування ріщень на базі ARDUINO.

У результаті роботи було побудовано декілька ресурсоефективних рішень з досяганням найменшої ціни на один виріб.

Повністю написано нову програму для отримання вимірів з декількох пристроїв на один сайт водночас. Описано шлях вдосконалення цих пристроїв.

ARDUINO, ESP8266, WI-FI, IoT, ІНТЕРНЕТ РЕЧЕЙ, DHT11

ABSTRACT

The purpose of the work is to create resource efficient IoT-solution for checking indicators of humidity and air temperature in a several rooms of thirty's building of Kyiv Polytechnic Institure.

We are looking at IoT as a technology of the future, its prospects and shortcomings.

Was buid step-by-step algorithm for creating and programming IoT solutions based on ARDUINO .

The simplest IoT solutions are reviewed, the algorithm of creating and programming solutions based on ARDUINO is described step by step.

As a result, several resource-efficient solutions were built with the lowest price per product.

Were written a completely new program to collect measurements from several devices on one site at the same time. The way of improvement of these devices is described

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

IoT	Internet of things – інтернет речей
IETF	Internt Engineering Task Force – Инженерна рада інтернету
ITU-T	International Telecommunication Union Telecommunication Standardization Sector - Міжнародний союз електрозв'язку, сектор телекомунікацій
IDC	International Data Corporation

ВСТУП

Про IoT говорять сьогодні мало не з кожної («розумної») праски. При цьому в таких розмовах зазвичай пропускають базові речі: що таке інтернет речей, з чого він складається, і хто може відповідати на ці питання на правах «уповноваженого органу». А між тим, питання ці дуже актуальні. Ось, скажімо, ваша «розумна» праска - відноситься до інтернету речей? Тут ми усе і дізнаємося.

Метою цієї роботи є знайомство з базисом інтернету речей та створення свого власного проекту із зчитування показників температури та вологості у аудиторіях тридцятого корпусу Київського Політехнічного Інституту імені Ігоря Сікорського.

Завдання дослідження:

1. Обгрунтування актуальності розробки власного IoT – рішення.
2. Побудова IoT - рішення на прикладі плати Elecrow ESP8266.
3. Модернізація рішення до ресурсоефективного.

Робота складається з чотирьох розділів:

У першому розділі було описано історію виникання інтернету речей як поняття. Наведено декілька визначень від різних видань або видатних людей. Досліджені перспективи та основні недоліки. Також було показано з чого він складається.

У другому розділі було обрано рішення від одного з найбільших виробників. Це рішення було повністю досліджене та описане. Також були описані основні інтерфейси обміну інформацією та живлення. Було створено алгоритм побудови та програмування цього рішення. Виявлені основні недоліки та переваги.

У третьому розділі було оглянуто концепцію IoT – платформ та описано для чого вони потрібні.

Повністю зроблений та показаний алгоритм з модернізації вже існуючого рішення. Також наочно показано як можно перепрограмувати свій

пристрій для взаємодії з платформою. У результаті чого показники можна виводити у смартфон.

Також показано результати клонування IoT – рішення, для передачі показників температури повітря і вологості з декількох аудиторій на сайт кафедри.

У четвертому розділі було оглянуте різноманіття датчиків у світі інтернету речей. Для тестів було узято датчик вогню, звуку та вологості ґрунту. Програма була змінена для опрацювання основного датчику температури та вологості разом з додатковим.

1 ОБГРУНТУВАННЯ АКТУАЛЬНОСТІ ТЕМИ ДОСЛІДЖЕННЯ

1.1 Концептуальні засади побудови системи

Дослідницька компанія Gartner визначає IoT як мережу фізичних об'єктів, що взаємодіють із зовнішнім середовищем і між собою, а також для передачі відомостей про свій стан.

Менш абстрактне визначення пропонує McKinsey: IoT - це датчики, вбудовані в фізичні пристрої та підключення до інтернету через дротові або бездротові мережі.

Розвитком IoT займаються не тільки виробники пристроїв, але і спеціалізовані організації, в числі яких Міжнародний союз електрозв'язку (ITU), Industrial Internet Consortium і IETF.

У рекомендаціях Y.2060 Міжнародного союзу електрозв'язку, які отримали назву Overview of the Internet of Things, інтернет речей постає як «глобальна інфраструктура, що надає складні послуги завдяки з'єднанню фізичних і віртуальних речей на основі існуючих і розвиваються функціонально сумісних інформаційно-комунікаційних технологій». Під річчю в цьому визначенні розуміється предмет фізичного або віртуального світу, який може бути ідентифікований та підключений до мереж зв'язку. Пристроєм в контексті IoT називається елемент обладнання, який володіє обов'язковими можливостями зв'язку і може проводити вимірювання, спрацьовувати при певних умовах, вводити, зберігати і обробляти дані.

Відповідно до рекомендацій ITU-T IoT є мережею пристроїв, тісно пов'язаних з речами. Сенсорні і виконавчі пристрої взаємодіють з фізичними речами в навколишньому середовищі. Пристрої збору даних зчитують інформацію з фізичних речей або записують її на фізичні речі, взаємодіючи з пристроями перенесення даних або носіями даних, підключеними або пов'язаними з фізичним об'єктом.

Іншими словами, IoT - це: Фізичні / Віртуальні об'єкти + контролери / сенсори / виконавчі механізми + інтернет

IOT - концепція простору, в якому все з аналогового і цифрового світів може бути поєднане - це перевизначить наші стосунки з об'єктами, а також властивості і суть самих об'єктів. © Роб Ван Краненбург.

Тобто Інтернет речей - це не просто безліч різних приладів і датчиків, об'єднаних між собою дротяними і бездротовими каналами зв'язку і підключених до мережі Інтернет, а це більш тісна інтеграція реального та віртуального світів, в якому спілкування виробляється між людьми і пристроями.

Передбачається, що в майбутньому «речі» стануть активними учасниками бізнесу, інформаційних і соціальних процесів, де вони зможуть взаємодіяти і спілкуватися між собою, обмінюючись інформацією про навколишнє середовище, реагуючи і впливаючи на процеси, що відбуваються в навколишньому світі, без втручання людини.

На думку Роба Ван Краненбург інтернет речей вдає із себе «чотирьох-шаровий пиріг».

1 рівень пов'язаний з ідентифікацією кожного об'єкта.

2 рівень надає з сервісом по обслуговуванню потреб споживача (можна розглядати як мережу власних «речей», приватний приклад - «розумний дім»).

3 рівень пов'язаний з урбанізацією міського життя. Тобто це концепція «розумного міста», де вся інформація, яка стосується жителів цього міста, стягується в конкретний житловий квартал, в Ваш будинок і сусідні будинки.

4 рівень - сенсорна планета.

Іншими словами Інтернет речей можна розглядати як мережу мереж, в якій невеликі мережі утворюють більші.

Далі приведено декілька прикладів для кожного типу IoT:

Розумний дім:

1) сервіси «Розумна квартира», підключені до системи збору даних (ОДС - об'єднана диспетчерська служба), інтегровані з системою IP-домофону дому та особистим кабінетом користувача;

2) безпеку квартири (миттєві повідомлення про несанкціонований доступ, протипожежна охорона, повідомлення про ризик затоплення та ін.);

3) зниження споживання ресурсів (індивідуальні лічильники тепла, води та електроенергії, контроль освітлення в приміщеннях, сервіси з аналізу споживання ресурсів);

4) турбота про дітей і літніх родичів (сервіси та мобільні додатки з безпеки дітей і літніх родичів, які перебувають в квартирі без нагляду).

Розумне місто:

1) рішення напрямку «безпека»: системи відеоспостереження та відеоаналізу, системи оповіщення та інформування, системи контролю якості й обігу ліцензованої продукції;

2) рішення по напрямку «керованість»: системи планування містобудівної діяльності, електронного документообігу та контролю доручень, системи моніторингу і контролю ЖКГ;

3) рішення по напрямку «комфорт»: комплекси моделювання та керування транспортними потоками, організація паркувального простору, державні (муніципальні) послуги, інформаційно-довідкові послуги.

Сільське господарство:

1) платформи з управління фермою, що поєднують систему сільськогосподарської техніки, зрошувальні системи (локальні датчики, іригаційні вузли, зрошувальні додатки);

2) системи метеоданих (датчики дощу, вологості, температури, метеозведення);

3) системи підготовки насіння (бази даних насіння, бази даних характеристик ферми, системи підготовки насіння);

Індустріальний інтернет речей:

1) виробництво інноваційних промислових датчиків, контролерів і модулів;

2) сенсори, передавачі та приймачі, точки доступу та обробки інформації, методи підвищення тривалості автономної роботи пристроїв.

Найголовнішою проблемою на сьогоднішній день є відсутність стандартів в цій галузі, що ускладнює можливість інтеграції пропонованих на ринку рішень і багато в чому стримує появу нових.

Так само для повноцінного функціонування такої мережі необхідна автономність всіх «речей», тобто датчики повинні навчитися отримувати енергію з навколишнього середовища, а не працювати від батарейок, як це відбувається зараз.

Наявність величезної мережі, яка контролює весь навколишній світ, глобальна відкритість даних та інші особливості можуть мати і негативні наслідки.

Ми вже можемо спостерігати появу розумних міст, розумних автомобілів, розумних фабрик і багатьох інших розумних об'єктів. Мобільні телефони та планшети вже давно підключені до Мережі - в них спочатку були встановлені всі необхідні для цього мікросхеми і модулі - проте мікрохвильові печі, холодильники і навіть одяг починають «входити» в інтернет тільки зараз.

І ось що цікаво: якщо в звичайну пару кросівок вбудувати систему датчиків, які б визначали, як використовується спортивне взуття, то зібрана інформація буде представляти цінність сама по собі. Зібравши достатню кількість даних, можна буде проаналізувати поведінку спортсмена, виявити помилки, поліпшити техніку бігу. Таким чином, звичайний атрибут нашої з вами життя - взуття - перетворюється в сервіс.

За прикладом ходити далеко не треба. Нові технології здатні вирішити безліч проблем охорони здоров'я, пропонуючи унікальні рішення для дослідження хвороб. Наприклад, компанії IBM і Pfizer об'єдналися для роботи над проектом, завдання якого дослідити хворобу Паркінсона. Датчики, встановлені на тілі хворого і в його будинку, дозволять визначити, як симптоми хвороби впливають на активність людини.

Сьогодні відстежити симптоми недуги Паркінсона досить складно, оскільки вони змінюються протягом дня, а лікарі спостерігають хворого лише обмежений період часу. У зв'язку з цим стеження за хворобою є досить складним випробування для пацієнтів, лікарів і дослідників.

Взагалі, західна публіка вельми делікатно ставиться до питань здоров'я. У 2014 році в США число регулярних відвідувачів спортзалів досягло рекордної позначки 54,1 мільйона чоловік (це 17% населення країни), а доходи американської фітнес-індустрії перевищили $ 22,4 млрд (це майже вдвічі перевищує показники 2000 року). Тому не дивно, що з'являються такі стартапи, як Naked Labs з Сан-Франциско, який почав приймати замовлення на систему 3D-сканування тіла.

Генеральний директор і співзасновник компанії Фархад Фарахбакхшіан (Farhad Farahbakhshian) говорить, що фітнес-трекер компанії розробляється для людей, які хочуть стежити за своїм прогресом при заняттях спортом. Девайс складається з великого дзеркала, що підключається до звичайної розетки, а також бездротовий майданчик для забезпечення рівномірного сканування. Користувачі отримують можливість переглядати 3D-скани на смартфоні: на екрані з'являється точна копія тіла людини з коментарями біля кожної дільниці тіла.

Іноді здається, що чудовий світ інтернету речей - не більше ніж фантастична картинка. Це не так. У доповіді Fortune Business Insights вказується, що світовий ринок Інтернету речей, вартість якого в 2018 році оцінювалася в 190 мільярдів доларів, досягне до 2026 року 1,11 трильйона доларів, продемонструвавши сукупний темп зростання 24,7% в рік.

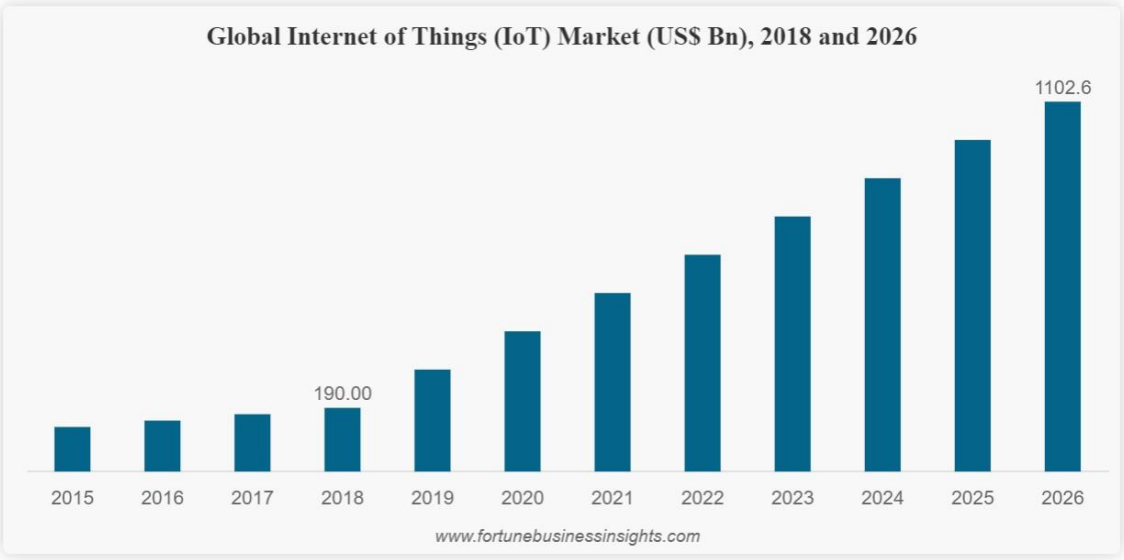

Рисунок 1.1 - «Прогноз ринку IoT в 2018-2026». Джерело: Fortune Business Insights

Очікується, що найбільшим сегментом ринку буде банківський сектор і сектор фінансових послуг.

Аналітики Gartner повідомляють, що в 2019 році кількість пристроїв IoT досягло 14,2 млрд. Компанія також прогнозує, що до 2025 року кількість підключених пристроїв досягне рівня в 25 мільярдів.

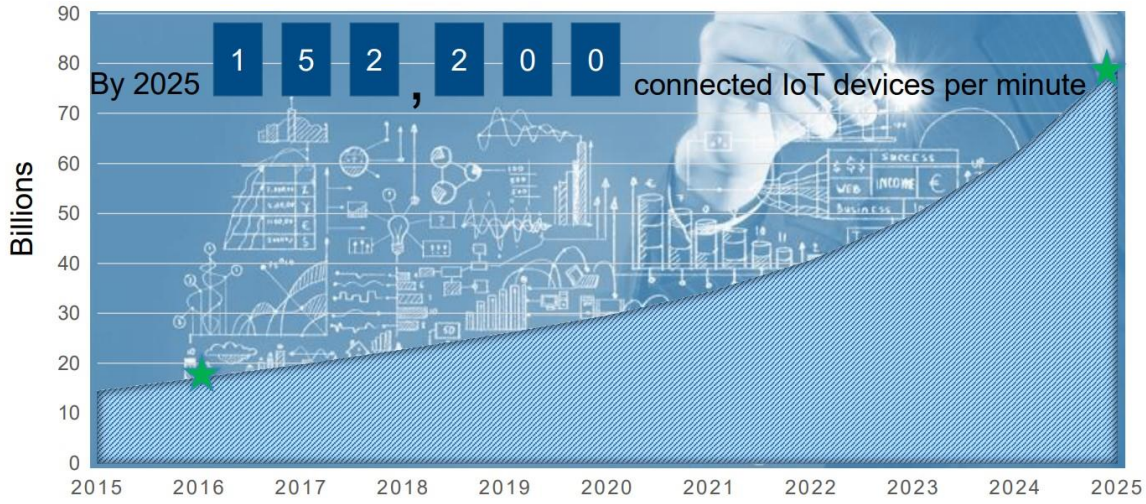

Рисунок 1.2 – «Прогноз IDC о зростанні кількості приладів IoT»

IDC дають ще більш оптимістичний прогноз: до 2025 року до мережі інтернету речей буде щохвилини підключатися 152 200 пристроїв.

Помноживши 152 200 на 525 600 (кількість хвилин в році), отримаємо, що в 2025 році інтернет речей буде містити близько 80 мільярдів пристроїв.

За даними дослідження IoT - The Internet of Transformation 2018, опублікованого Juniper Research, ключовими ринками IoT залишаються Північна Америка, Західна Європа, Далекий Схід і Китай. Саме ці регіони забезпечать більш 60% всіх доходів, пов'язаних з інтернетом речей.

1.2 Архітектура IoT

Сенсори:

Це прилади для зняття тих чи інших зовнішніх показників і передавання цих показників для подальшої обробки. Сенсорів існує велике ріноманіття, починаючи від стандартних термометрів, мікрофонів, камер, та закінчуючи десятками менш розповсюджених.

Деякі з них можна побачити на Рис.1.3 «Sensors Starter Kit для Arduino».

Рисунок 1.3 - «Sensors Starter Kit для Arduino».

Актуатори:

Даний тип елементів призначається для того, щоб впливати на навколишнє середовище, або на певний об'єкт в ній. Цю роль можуть виконувати найрізноманітніші пристрої: від сервоприводів і динаміків до замків (звичайно, електронних) з освітлювальними приладами.

Гейти:

Це пристрої, на які зазвичай покладають логіку поверхневого аналізу інформації, що надходить від підключених до них сенсорів. У певних ситуаціях, аналіз даних може вимагати малої кількості обчислювальних ресурсів, так що гейти цілком здатні приймати деякі рішення самостійно. Беручи такі рішення, вони відправляють певні команди управління на актуатори, які, в свою чергу, виконують вже свої функції.

Якщо ж обробка иформации вимагає великих витрат, або ця інформація підлягає збору, гейти відправляють її на сервера, де з нею і проводиться подальша робота. Цілком собі ймовірно використання в ролі гейтов мікрокомп'ютерів (Рис.1.4) або мікропроцесорів (Рис.1.5):

Рисунок 1.4 – «Мікрокомп’ютер»

Рис.1.5 – «Мікропроцесор»

Для того, щоб побудувати моніторингову систему, досить буде використання лише сенсорів і деякого сервера, який буде виступати в ролі гейта. Наприклад, завдяки сенсору руху і звичайній платі, можна без особливих зусиль організувати облік кількості людей, що проходять через якусь прохідну.

Тепер, розглянувши пристрої мережі інтернету речей, можна точно сказати, що в плані апаратної частини немає нічого загадкового і складного. Зробити простеньку IoT-мережу може будь-хто, здатний купити досить дешеві на сьогоднішній день компоненти і написати код з пари рядків. Однак для того, щоб розробити і втілити у життя серйозні проекти як, наприклад, реалізацію концепції розумного будинку або навіть розумного міста, потрібно докласти величезну кількість зусиль. Адже для того, щоб всі ці пристрої працювали між собою потрібна платформа, здатна контролювати всі протікають процеси.

Так само не варто забувати, що в хмарах інтернету речей можуть використовуватися і інші технології, які допомагають розкрити його потенціал в більшій мірі. Такими можуть виступати і BigData, і BlockChain, і нейромережі з машинним навчанням. Але ж кожна з останніх перерахованих

технологій являє собою окрему велику область комп'ютерних (і не дуже) наук.

1.3 Постановка завдання дослідження

Як було написано у п.1.1, IoT все більше втручається у наше життя і його неможливо оминути стороною. Інтернет речей – це технологія, що поліпшує життя людей.

Про інтернет речей знають багато людей, але майже усі з них вважають, що це складно і занадто дорого. Але це не так.

Головне завдання цього дослідження зробити пристрій для отримання показників температури та вологості і в подальшому розмноження цих пристроїв для отримання показників з декількох кімнат, аудиторій або будівель.

Також ця робота присвячена знайомству з інтернетом речей – його найпростішим аспектам.

У цій роботі буде повністю показано алгоритм створення міні-проекту, за яким може кожний зробити щось своє на власний розсуд.

1.4 Висновки з розділу 1

1. Інтернет речей - це не просто безліч різних приладів і датчиків, об'єднаних між собою дротяними і бездротовими каналами зв'язку і підключених до мережі Інтернет, а це більш тісна інтеграція реального та віртуального світів, в якому спілкування виробляється між людьми і пристроями.
2. Передбачається, що в майбутньому «речі» стануть активними учасниками бізнесу, інформаційних і соціальних процесів, де вони зможуть взаємодіяти і спілкуватися між собою, обмінюючись інформацією про

навколишнє середовище, реагуючи і впливаючи на процеси, що відбуваються в навколишньому світі, без втручання людини.

3. Найголовнішою проблемою на сьогоднішній день є відсутність стандартів в цій галузі, що ускладнює можливість інтеграції пропонованих на ринку рішень і багато в чому стримує появу нових.

4. Для того, щоб побудувати моніторингову систему, досить буде використання лише сенсорів і деякого сервера, який буде виступати в ролі гейта. Наприклад, завдяки сенсору руху і звичайній платі, можна без особливих зусиль організувати облік кількості людей, що проходять через якусь прохідну.

2 ПОБУДОВА РЕСУРСОЕФЕКТИВНОГО IoT – РІШЕННЯ НА ПРИКЛАДІ ПЛАТИ ELECROW ESP8266

2.1 Огляд ресурсоефективного рішення

Для знайомства з IoT було придбано рішення від компанії Elecrow під назвою ESP8266 IoT Weather Station Kit(Рис.2.1)

Комплект містить:

1) Плата ESP8266 2 шт.
2) 30см micro USB кабель 2 шт.
3) OLED Екран
4) Сенсор дощу
5) Сенсор ультрафіолетового випромінювання
6)Сенсор температури та вологості
7)Кабель для підключення сенсорів та екрану 4 шт.

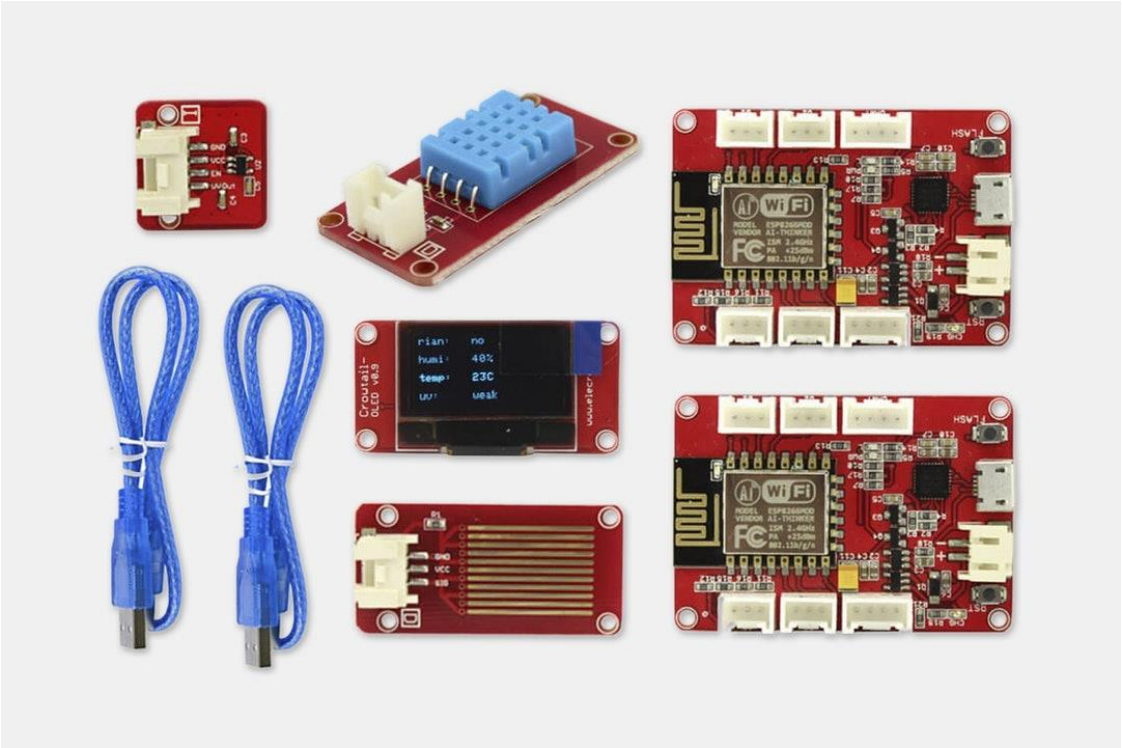

Рисунок 2.1 – «ESP8266 IoT Weather Station Kit»

Розглянемо плату ближче:

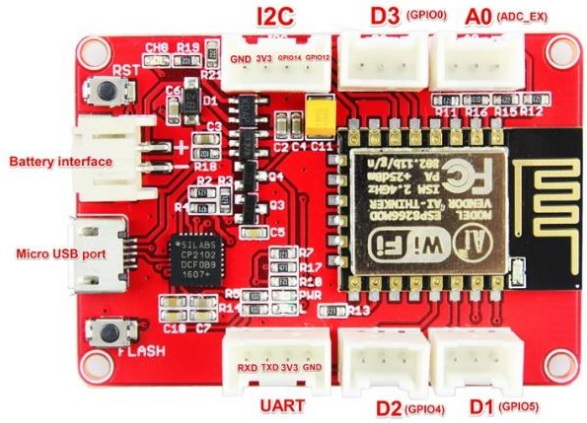

Рисунок.2.2 – «Плата з комплетку ESP8266 IoT Weather Station Kit»

Модифікована плата ESP8266(Рисунок 2.2) має розміри 50ммх35мм, з частотою процессору 80 Мегагерц та потребує живлення з напругою від 3.3 до 5 вольт, та силою струму до 0.5 Ампер. Є вбудований WI-FI модуль, що полегшує програмування та процедуру з'єднання плат між собою

Плата має 6 інтерфейсів підключення:

1) Battery interface – інтерфейс підключення живлення від батареї або акумулятору. Використовується, коли потрібна мобільність приладу і свобода від розеток.

2) Micro USB port – інтерфейс підключення до комп'ютера для завантаження програми на плату, також може використовуватися для підключення блоку живлення від розетки.

3) I2C – інтерфейс для передачі інформації, також слугує для з'єднання плат між собою.

4)UART – інтерфейс вводу та виводу цифрової інформації

5)D1,D2,D3 – інтерфейси вводу цифрового сигналу, слугують для ідключення цифрових датчиків.

6)A0 – інтерфейс вводу аналогового сигналу, слугує для підключення аналогових датчиків

Якщо загалом більшість інтерфейсів нам зрозуміла, то UART і I2C багато хто з нас бачить вперше, а отже, краще буде призупинитися на цих двух інтерфейсах.

UART розшифровується як універсальний асинхронний прийом і передача і являє собою простий протокол зв'язку, який дозволяє Arduino спілкуватися з іншими пристроями.

Цей інтерфейс, який знаходиться на всіх платах Arduino, дозволяє Arduino безпосередньо спілкуватися з комп'ютером завдяки тому, що Arduino має вбудований перетворювач USB. Тому програми, написані на ОС Windows, Mac або Linux, можна використовувати з Arduino, підключеним до порту USB.

I2C (inter-integrated-circuit) - це протокол послідовного зв'язку, спеціально розроблений для мікроконтролерів.

Хоча цей периферійний апарат майже ніколи не використовується для зв'язку між ПК та пристроєм, він користується надзвичайно популярною модулями та датчиками, що робить його корисним для проектів, які потребують спільної роботи багатьох частин. Насправді I2C дозволяє потенційно підключити до 128 пристроїв до вашої основної плати!

I2C дає можливість підключити декілька «MASTER» і «SLAVE» до вашої плати, зберігаючи чіткий шлях зв'язку.

Підтримувати чіткий шлях зв'язку можливо, оскільки I2C використовує адресну систему та загальну шину, тобто багато пристроїв можуть бути підключені до тих самих дротів.

Також плата мітить 2 кнопки, перша (RST) слугує для повного видалення програми з пам’яті, а друга (FLASH) потрібна для оновлення програмного забезпечення.

Загалом цей комплект має великий потенціал використовування. Починаючи програмою від виробника (знімати з одної плати та передавати на іншу в одній мережі WI-FI) та закінчуючи модифікаціє цієї програми для

передачі інформації на велику відстань на цей же екран або на сайт чи телефон.

Коплект зроблено якісно, дефектів помічено не було. Навіть після падіння плата працювала далі і ніяких пошкоджень не отримала.

2.2 Побудова та програмування ресурсоефективного рішення

Щоб запрограмувати плату треба виконати декілька простих кроків:

1. Завантажити програму Arduino IDE з офіційного сайту (https://www.arduino.cc/en/Main/Software)
2. Встановлюємо програму
3. Запустити програму та увійти у «Файл/Настройки» (Рисунок2.3)

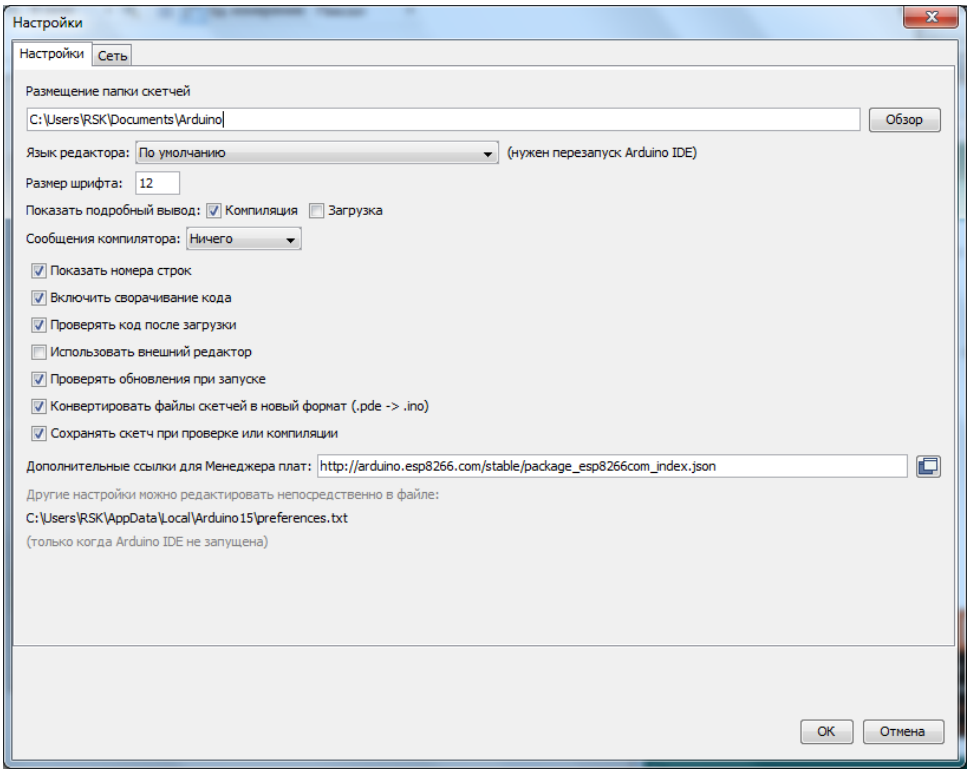

Рисунок 2.3 – «Налаштування програми Arduino IDE»

4. Вставити силку (http://arduino.esp8266.com/stable/package_esp8266com_index.json) у поле «Дополнительные ссылки для Менеджера плат»

5. Переходимо до «Инструменты/Плата:/Менеджер плат» та скролимо його до самого низу і бачимо (Рис.2.4)

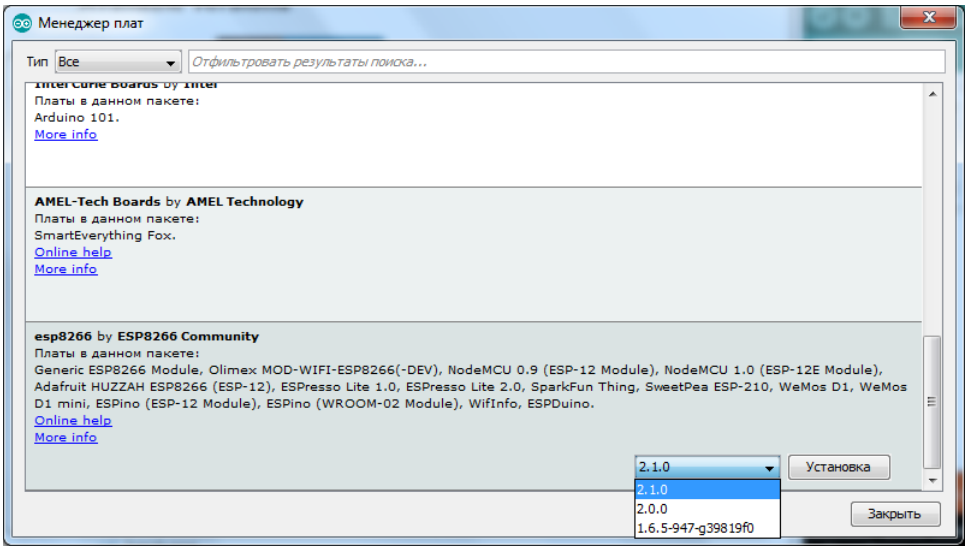

Рисунок.2.4 – «Менеджер плат»

6. Встановлюємо «esp8266 by ESP8266 Community»

7. Далі йдемо у «Инструменты» і у полі Arduino/Genuino Uno обираємо Generic ESP8266 Module(Рис.2.5) |

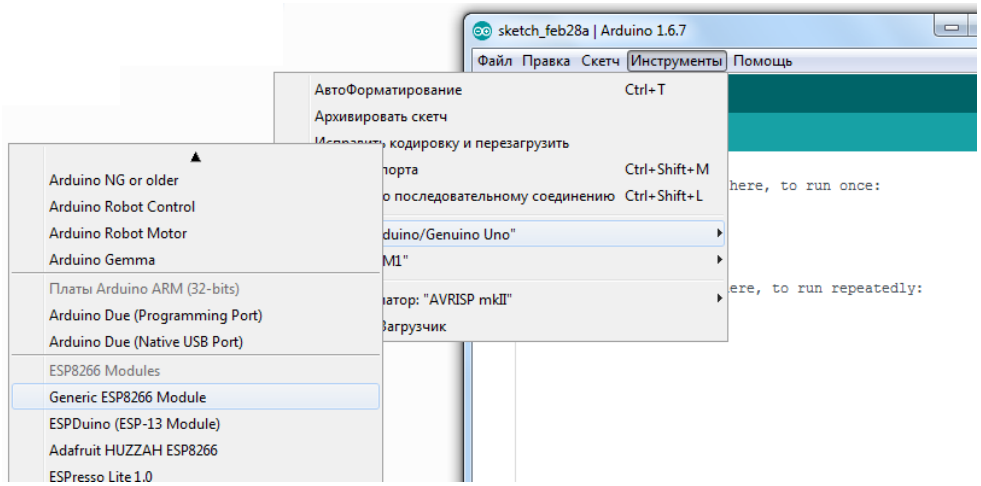

Рисунок.2.5 – «Інструменти»

8. Далі вказуємо потрібний COM порт та ставимо Upload Speed 115200(Рис.2.6)

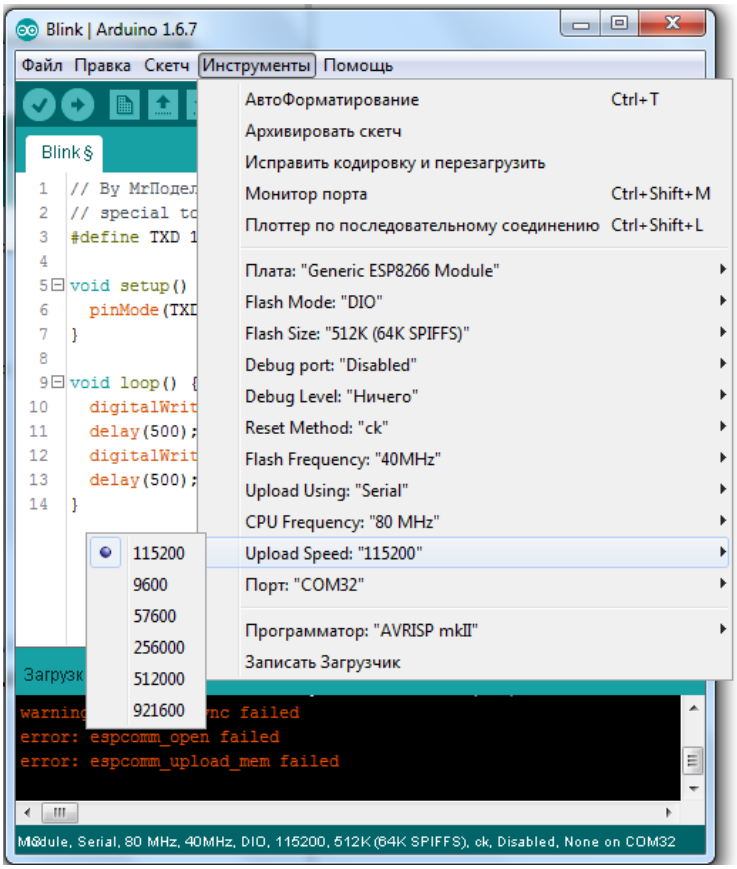

Рисунок 2.6 – «Встановлення швидкості»

9. Завантажуємо з сайта виробника два файли з назвою «A»(Додаток А) та «B»(Додаток Б)

10. Далі під’єднуємо плату «A» (з сенсорами)

11. У пусте поле програми вставляємо код з завантаженого файлу «A»

12. Змінюємо «Elecrow803» на назву точки доступу WI-FI, та «elecrow2014» на пароль від точки доступу.

```
const char *ssid     = "Elecrow803";//ere is my wifi, modified to you
const char *password = "elecrow2014";//wifi password
```

13. Нажимаємо стрілочку щоб завантажити програму на плату.

14. Далі переходимо у налаштування WI-FI у браузері(зазвичай 192.168.0.1 або 192.168.1.1) та у пункті «Пристрої» знаходимо плату і запам’ятовуємо IP

15. Під'єднуємо плату «B» та повторюємо пункт 12 але з файлом «B»

16. Змінюємо назву WI-Fi та пароль як у п.13 та вписуэмо IP, який запам’ятали раніше у поле «host»

```
const char *ssid     = "Elecrow803";//
const char *password = "elecrow2014";
const char *host = "192.168.1.184";//
```

17. Перезавантажуємо обидві плати

2.3 Висновки до розділу 2

1. Для знайомства з IoT було придбано рішення від компанії Elecrow під назвою ESP8266 IoT Weather Station Kit(Рис.2.1)

2. UART розшифровується як універсальний асинхронний прийом і передача і являє собою простий протокол зв'язку, який дозволяє Arduino спілкуватися з іншими пристроями.

3. I2C (inter-integrated-circuit) - це протокол послідовного зв'язку, спеціально розроблений для мікроконтролерів.

4. Загалом цей комплект має великий потенціал використовування. Починаючи програмою від виробника (знімати з одної плати та передавати на іншу в одній мережі WI-FI) та закінчуючи модифікаціє цієї програми для передачі інформації на велику відстань на цей же екран або на сайт чи телефон.

5. Наочно показан алгоритм програмування.

3 МОДЕРНІЗАЦІЯ РЕСУРСОЕФЕКТИВНОГО РІШЕННЯ

3.1 Передача показників на сервер Blynk

Щоб вивести IoT- рішення далі власної WI-FI мережі, потрібно буде звернутися до IoT-платформи.

IoT-платформа - це рішення, що забезпечують уніфіковану взаємодію між кінцевими пристроями IoT і сервісами, обробними дані. А пояснювати, чому вони важливі, почнемо здалеку.Дослідження Cisco виявило, що 75% проектів, пов'язаних з IoT, зазнають невдачі. В опитуванні взяли участь понад 1800 керівників компаній та IT-лідерів, метою опитування було виявлення основних бар'єрів, що обмежують впровадження інтернету речей на підприємствах. Згідно з висновками дослідження, основними перешкодами для організацій, які бажають впровадити IoT, стають витрати і терміни реалізації проектів. Ще одним стоп-фактором стала обмеженість експертних знань штатних співробітників.

Усунути ці проблеми дозволяє використання рішень, що забезпечують уніфіковане взаємодія між кінцевими пристроями IoT і сервісами, обробними дані, - тих самих IoT-платформ.

Пояснимо: якщо в компанії вже є парк обладнання, при впровадженні IoT потрібно підключити його до нової інфраструктури. При цьому якась частина «старих» пристроїв може цілком успішно виконувати свої виробничі функції, але не мати можливості підключення до інтернету. Заміна такого обладнання на IoT-сумісний спричинить великі витрати. Це збільшить термін окупності, оскільки доведеться списати цілком працездатні верстати і агрегати.

Але навіть якщо обладнання сумісно з IoT, залишається відкритим питання з тим, які дані необхідно збирати і використовувати, як проводити поглиблений аналіз зібраної інформації і забезпечити оперативний зворотний зв'язок. IoT-платформи якраз і забезпечують безшовну інтеграцію апаратних

засобів з використанням різних типів підключення, передачу даних на підключені пристрої або між ними.

IoT-платформи пропонують багато високотехнологічні та IT-компанії. Розробка компанії Toshiba для інтеграції IoT-пристроїв і сервісів отримала назву SPINEX. При розробці IoT-платформи SPINEX використовувався великий досвід Toshiba в енергетиці, виробництві напівпровідникових компонентів, а також в області інтернету речей, штучного інтелекту, розпізнавання голосу і відео.

Програмування Arduino або Raspberry Pi – одне з найбільш цікавих занять двадцятого століття. Побудова пристроїв на їх основі давно вийшло за рамки недосяжного для звичайної людини. Вони використовуються для створення верстатів, роботів, 3-D принтерів, квадрокоптерів, серверів та IoT-пристроїв.

Нажаль, цей ринок сильно сегментований. Вони програмуються у різних середовищах та через різні інтерфейси. Цю ситуацію може врятувати проект Blynk.

Blynk – це хмарний сервіс для створення графічного інтерфейсу, через який буде керуватися ваш мікрокомп'ютер або мікроконтролер. Якщо раніше треба було писати дуже складний інтерфейс для збору інформації з датчиків, то зараз можна обійтися роботою в Blynk, і це займе набагато менше часу.

Щоб створити власний проект через Blynk треба всього нічого: встановити додаток та зареєструватися в один крок. Реєстрація обов'язкова, бо Blynk – хмарний сервіс, і доступ до нашого пристрою може отримати будь хто.

Але ми запустимо сервер локально. Отже і доступ до інтернету нам не знадобиться.

Для запуску та управління програмою потребуються деякі навички. Спочатку треба пов'язати мікрокомп'ютер або мікроконтролер з комп'ютером. Blynk дозволяє від'єднувати найрізноманітніші інтерфейси:

-Official Ethernet Shield (W5100)

- Adafruit CC3000 WiFi

-ENC28J60

-RN-XV WiFly

-ESP8266

-USB (Serial)

-SeeedStudio Ethernet Shield V2.0 (W5200)

-Official Arduino WiFi Shield

-Official Ethernet Shield (W5100)

-ESP8266 (WiFi modem)

Платформа підійде як новачкам, так і більш просунутим користувачам, які не хочуть витрачати час на написання додатків для управління проектами: від зчитування даних з метеостанції і управління розумним будинком до управління роботами.

Вся інформація, необхідна для початку роботи, розміщена на офіційному сайті. Blynk - це open-source-проект, так що кожен може взяти участь у створенні нових функцій. На даний момент використання сервісу повністю безкоштовно, надалі ситуація дещо зміниться - перш за все, за рахунок монетизації нових функцій. Так, вже зараз відомо, що доступ до GPIO-інтерфейсів буде купуватися як вбудована покупка.

На даний момент Blynk працює з наступними платами:

-ESP8266;

-Raspberry Pi;

-Arduino;

-Wicked WildFire (CC3000);

-Particle (ex Spark Core);

-TinyDuino (CC3000).

Щоб запустити Blynk – сервер локально треба йти у певній послідовності:

1. Скачати програму сервера з офіційного сайту https://github.com/blynkkk/blynk-server/archive/master.zip

2. Скачати потрібний jar – файл з офіційного сайту https://github.com/blynkkk/blynk-server/releases

3. Розархівувати у директорії скачування та у папці зробити папку data

4. Пройти до цієї директорії у командній строці (CMD)

5. Запустити сервер командою java -jar server/server-0.41.11-java8.jar -dataFolder data у командній строці

6. Далі під'єднуємося до серверу по цій ссилці https://10.30.2.33:9443/admin і бачимо наступну картину – сервер працює(Рис.3.1)

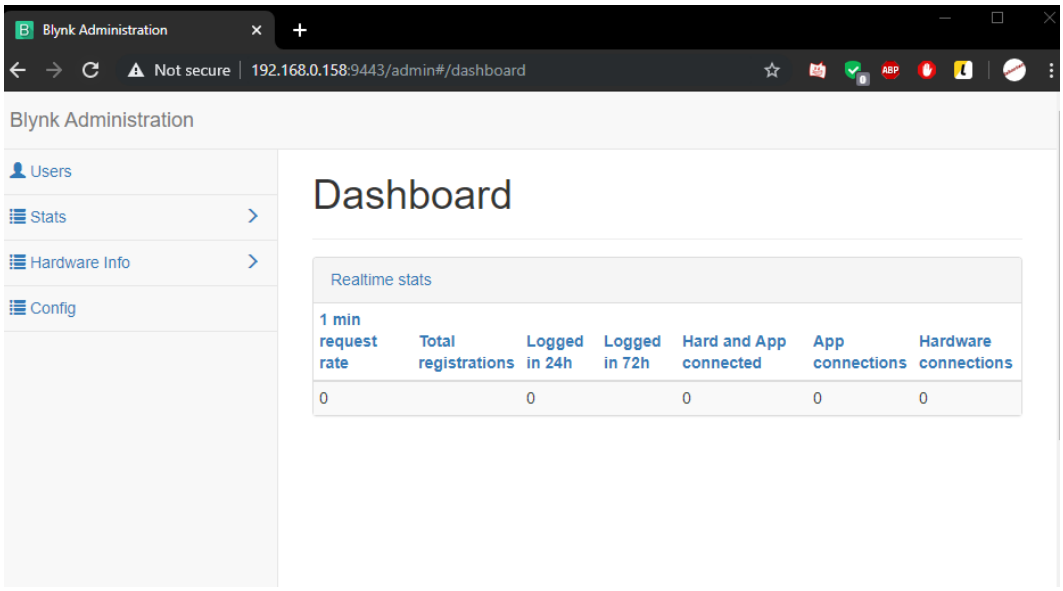

Рисунок 3.1 – «Blynk - сервер»

7. Далі добавити до вже існуючої програми код для коректної роботи з Wi-FI:

```
#include <ESP8266WiFi.h>
const char* ssid = "TS_ITS";
const char* password = "KafTSITS";
```

```
void setup(void)
{
 Serial.begin(115200);
 WiFi.begin(ssid, password);
 Serial.print("Connecting WIFi AP: ");
 while (WiFi.status() != WL_CONNECTED) {
 delay(1000);
 Serial.print(".");
 }
 Serial.println("");
 Serial.println("WiFi connected");
 Serial.println(WiFi.localIP());
}
void loop() {
}
```

3.2 Передача показників на сайт кафедри

Для початку модифікації було розроблено повну схему роботи.(Рис 3.2)

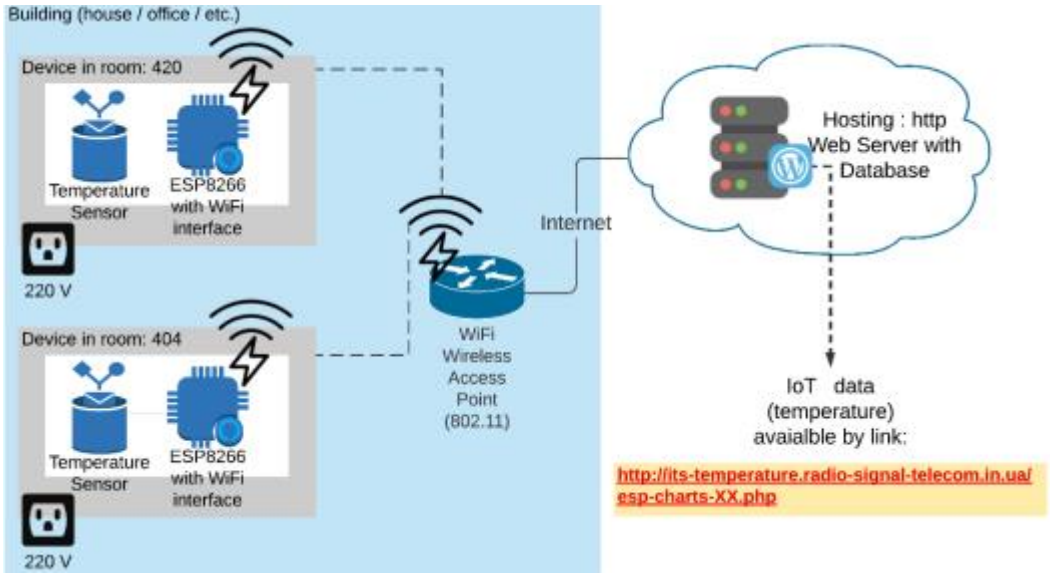

Рисунок 3.2 – «Схема роботи проекту»

Після розробки повної схеми було зрозуміло, що нам потрібно модифікувати програму та створити хост для прийому інформації, та сайт для побудови графіків.

Програма і код хосту та сайту знаходяться у Додатку 3. Алгоритм з програмування плати завжди однаковий, отже повторювати його тут не будемо. Варто зазначити, що загалом код майже не змінився, головною відмінністю є рядок «const char* serverName = "http://its-temperature.radio-signal-telecom.in.ua/temp_post_data.php";», що визначає місце, куди ми відправляємо дані.

Далі для створення хосту достатньо звернутися до сайту «https://github.com/», на якому можно знайти усе, що потрібно для будь-якої мови програмування. Там ми і беремо найпростіший код для нашого хосту.

Для сайту робимо так само, але є моменти, які треба змінити. Базового знання мови PHP буде достатньо.

Загалом код сайту виглядає великим і складним, але це не так.

Його основою є початок і кінець, який дублікується в стільки разів, скілбки у нас пристоїв. Сам сайт «http://its-temperature.radio-signal-telecom.in.ua/esp-charts-XX.php»(Рис.3.3) дуже простий і показує тільки графіки зміни температури в аудиторіях, у яких встановлені IoT прилади.

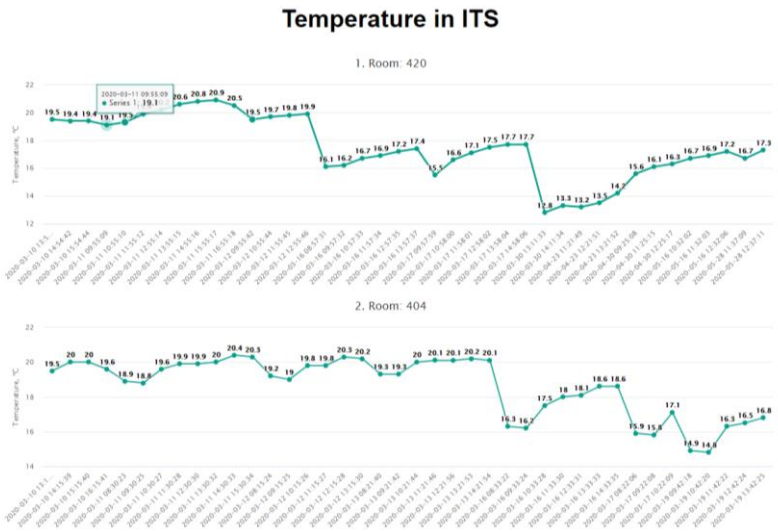

Рисунок 3.3 Графіки зміни температури в аудиторіях, у яких встановлені IoT прилади

3.3 Побудова більшої кількості пристроїв ресурсоефективного IoT рішення

Отже, для побудови більшої кількості пристроїв нам знадобиться більша кількість плат та датчиків.

За основу цього проекту було обрано датчик температури та вологості. Але, рішення, що булу куплено спочатку має надто високу ціну, від 45$ до 50$ США або 1300 гривень. Окремо така сама плата від цього ж виробника (Elecrow) коштує вже 15$ або 330 гривень (https://arduino.ua/prod1581-plata-razrabotchika-na-esp8266).

Отже, було обрано іншу плату(Рис.5.1)

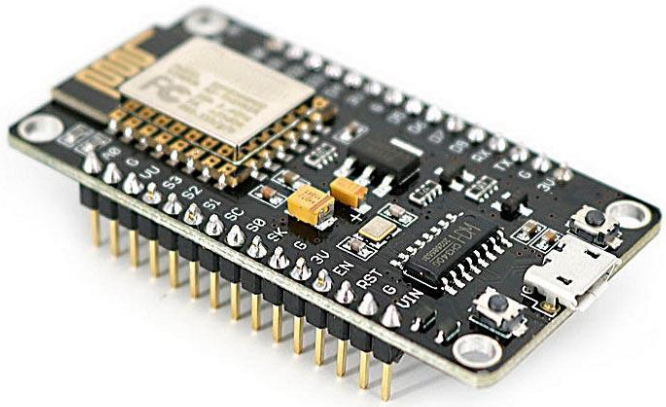

Рисунок 5.1 – «WIFI Модуль для Ардуіно NodeMcu Lua на ESP8266 CH340»

Це – WIFI Модуль для Ардуіно NodeMcu Lua на ESP8266 CH340. (https://erg.com.ua/p786347412-wifi-dlya-arduino.html?gclid=Cj0KCQjw-_j1BRDkARIsAJcfmTGcV8NVTX-fJ2qC-A3li5JU_eVvlhtfaA8bDuUt1JPAVcj28_LRnYAaAjtqEALw_wcB)

Функціонально плата не відрізняється від попередньої. Основної відмінністю можна охарактеризувати наявність вільно стоячих пінів.

Її ціна – 120 гривень.

Також нам потрібна буде більша кількість датчиків, для цього візьмемо аналог датчика, що був у комплекті “DHT11”(Рис.5.2).

Але, є більш точні датчики(Рис.5.3), але їх ціна для нас не підходить, оскільки вони коштують майже як ціла плата, а саме – 110грн (https://arduino.ua/prod1683-datchik-vlajnosti-i-temperatyri-dht21am2301).

Наш вибір –

датчик вологості і температури DHT11(https://arduino.ua/prod602-modyl-datchika-vlajnosti-i-temperatyri-dht11). Для знімання з нього даних не треба буде міняти код у програмі і його ціна всього 38 грн.

Рисунок 5.2 – «Датчик температури та вологості DHT11»

Рисунок 5.3 – «Датчик температури та вологості DHT21»

Для живлення на відстані від комп’ютера будемо використовувати старий блок живлення від смартфона з основними характеристиками: Напруга у 5 Вольт, та потужністю струму у 0.5 Ампера.

Корпус можна побудувати з будь-якої пластикової коробки, у якій було просверлено отвори для циркуляціїї повітря(Рис.5.4 та Рис.5.5). Також можна використати будь-який пластиковий контейнер з під риби, що знизить вартість «корпусу» приладу до 0 грн. Наш корпус (Рис.5.4 та Рис.5.5) коштував 20 грн.

Рисунок 5.4 – «Зібраний у корпусі проект»

Загальна ціна компонентів зазначена у Таблиці 1

Таблиця 5.1 – «Ціна компонентів проекту»

Назва	**Ціна**
Плата	120,00
Датчик	38,00
Дроти	25,00
Блок живлення	100,00

Отже, загальна ціна виробу 283 гривні.(Рис.5.5)

Рисунок 5.5 - «Зібраний у корпусі проект»

3.4 Висновки до розділу 3

1. IoT-платформа - це рішення, що забезпечують уніфіковану взаємодію між кінцевими пристроями IoT і сервісами обробки даних.
2. Blynk – це хмарний сервіс для створення графічного інтерфейсу, через який буде керуватися ваш мікрокомп'ютер або мікроконтролер. Якщо раніше треба було писати дуже складний інтерфейс для збору інформації з датчиків, то зараз можна обійтися роботою в Blynk, і це займе набагато менше часу.
3. Для створення хосту достатньо звернутися до сайту «https://github.com/», на якому можно знайти усе, що потрібно для будь-якої мови програмування. Там ми і беремо найпростіший код для нашого хосту.
4. Для сайту робимо так само, але є моменти, які треба змінити. Базового знання мови PHP буде достатньо.
5. Сам сайт «http://its-temperature.radio-signal-telecom.in.ua/esp-charts-XX.php»(Рис.3.3) дуже простий і показує тільки графіки зміни температури в аудиторіях, у яких встановлені IoT прилади.

6. Основою для ресурсоефективного рішення буде WIFI Модуль для Ардуіно NodeMcu Lua на ESP8266 CH340. Функціонально плата не відрізняється від попередньої. Основної відмінністю можна охарактеризувати наявність вільно стоячих пінів.Її ціна – 120 гривень.

7. Загальна ціна виробу 283 гривні.

4 ОГЛЯД І ПРОГРАМУВАННЯ НОВИХ ДАТЧИКІВ

Оскільки плата ESP8266 має можливість під’єднання великої кількості датчиків(три цифорових та один аналоговий), ми можемо використати це у своєму дослідженні.

Так існує велике різноманіття датчиків, нижче приведені усі основні категорії датчиків:

1. Датчики звуку та ультразвукові датчики приближення
2. Датчики освітленості та ультрафіолетового випромінювання
3. Датчики вогню
4. Датчики температури та вологості
5. Датчики – акселерометри та гіроскопи
6. Датчики вимірювання сили струму або напруги
7. Датчики газу або диму
8. Датчики дощу
9. Датчики вимірювання PH та для хімічного аналізу

До нашої плати можна під’єднати усі ці датчики. Вони можуть бути від одного виробника(Рис.4.1), у нашому випадку це Elecrow, так і від інших виробників.

Звісно ціни в усіх виробників різні, але аналогові датчики дешевше, що підходить для ресурсоефективного рішення, їх ми і будемо досліджувати .

Рисунок 4.1 – «Датчики для Arduino»

4.1 Огляд нових датчиків

Для тестів було обрано три датчики:

1. Датчик вогню від DFRobot (117 грн.)
2. Сенсор звуку аналоговий від DFRobot (156 грн.)
3. Ємкісний датчик вологості ґрунту (77 грн)

Усі датчики були куплені на одному з найбільших сайтів України, що спеціалізується на інтернету речей - Arduino.ua (https://arduino.ua/).

Датчик вогню(Рис.4.2) може вловлювати вогонь або інше випромінювання з довжиною хвилі 760 - 1100нм. Робочий кут складає 60 градусів. Так само на платі є два отвори під гвинти для фіксації. Робоча температура датчика від -25 до 85 градусів Цельсія, тому треба бути обережним, щоб не наближати датчик занадто близько до вогню та не

використовувати у занадто холодному просторі.(https://arduino.ua/prod299-Datchik_ognya).

Даташит до цього датчика по цій ссилці

https://arduino.ua/docs/FlameSensor.pdf

Рисунок 4.2 – «Датчик вогню»

Сенсор звуку (Рис.4.3) представляє з себе мікрофон зі стократно операційним підсилювачем, який може посилити звуки голосу, стуку в двері та ін, також може використовуватися для вимірювання гучності. Даташит до цього сенсора знаходиться тут https://github.com/Arduinolibrary/DFRobot_Sound_Sensor/blob/master/HYLD%209767%20Specification.pdf?raw=true

Рисунок 4.3 – «Датчик звуку»

Найбільша проблема резистивних датчиків вологості грунту - малий термін експлуатації, обумовлений схильністю корозії контактів вимірювача. Ємнісні датчики (Рис.4.4) вільні від цього недоліку, а корозійностійке покриття електродів робить їх практично вічними. Ще це позитивно впливає на стабільність показань і точність вимірювання датчика. Так само на стабільність показань позитивно впливає наявність на платі датчика стабілізатора напруги живлення, що дає можливість живити датчик напругою від 3,3 до 5,5В.

Рисунок 4.4 – «Датчик вологості грунту»

4.2 Програмування та зняття показників нових датчиків

Оскільки датчик звуку та датчик вогню аналогові, то і програмування для них однакове. Використаємо найпростіший код:

```
void setup(){
Serial.begin(9600); // Відкриваємо з’єднання з послідовним на 9600 біт
```

```
}
void loop(){
int val;
val=analogRead(0); //зчитуємо значення з аналогового порта 0
Serial.println(val,DEC);//виводимо значення на екран
delay(100);
}
```

Робота сенсору звуку та датчику воню показана на Рис.4.5 і Рис.4.6 відповідно

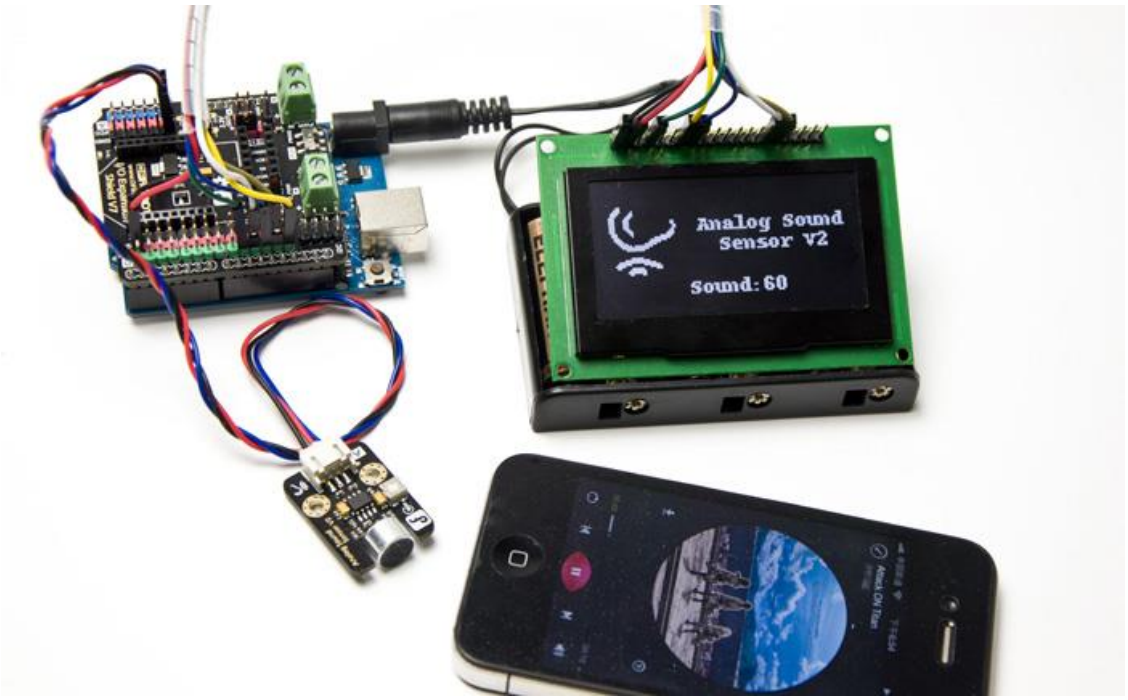

Рисунок 4.5 – «Робота датчику звуку»

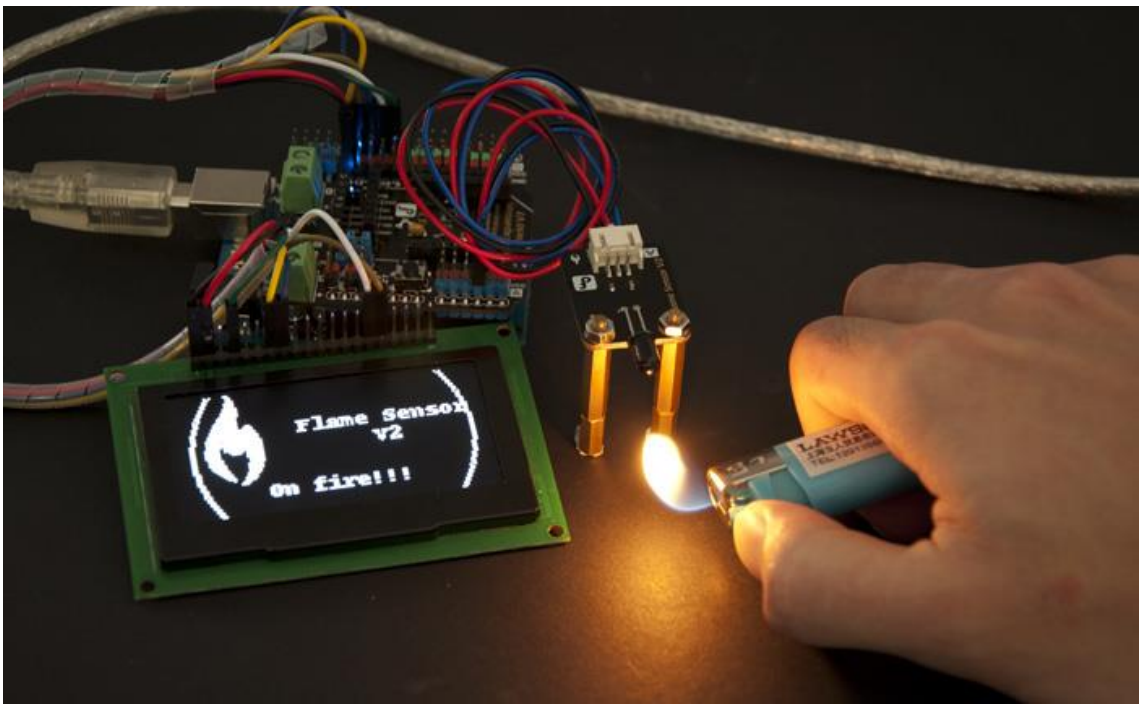

Рисунок 4.6 – «Робота датчику вогню»

Хоч ємкісний датчик вологості грунту і аналоговий, але логіка його роботи інша, отже і програма буде інша. Використаємо найпростіший код з сайту виробника по ссилці https://wiki.dfrobot.com/Capacitive_Soil_Moisture_Sensor_SKU_SEN0193.

```
const int AirValue = 520;   //you need to replace this value with Value_1
const int WaterValue = 260;  //you need to replace this value with Value_2
int intervals = (AirValue - WaterValue)/3;
int soilMoistureValue = 0;
void setup() {
  Serial.begin(9600); // open serial port, set the baud rate to 9600 bps
}
void loop() {
soilMoistureValue = analogRead(A0);  //put Sensor insert into soil
if(soilMoistureValue > WaterValue && soilMoistureValue < (WaterValue + intervals))
{
  Serial.println("Very Wet");
}
else if(soilMoistureValue > (WaterValue + intervals) && soilMoistureValue < (AirValue - intervals))
{
  Serial.println("Wet");
}
else if(soilMoistureValue < AirValue && soilMoistureValue > (AirValue - intervals))
{
  Serial.println("Dry");
}
delay(100);
}
```

Отже, виходячи з цього коду сухий грунт буде давати значення (520-430] включно, вологий (430-350], вода (350-260] (Рис.4.7)

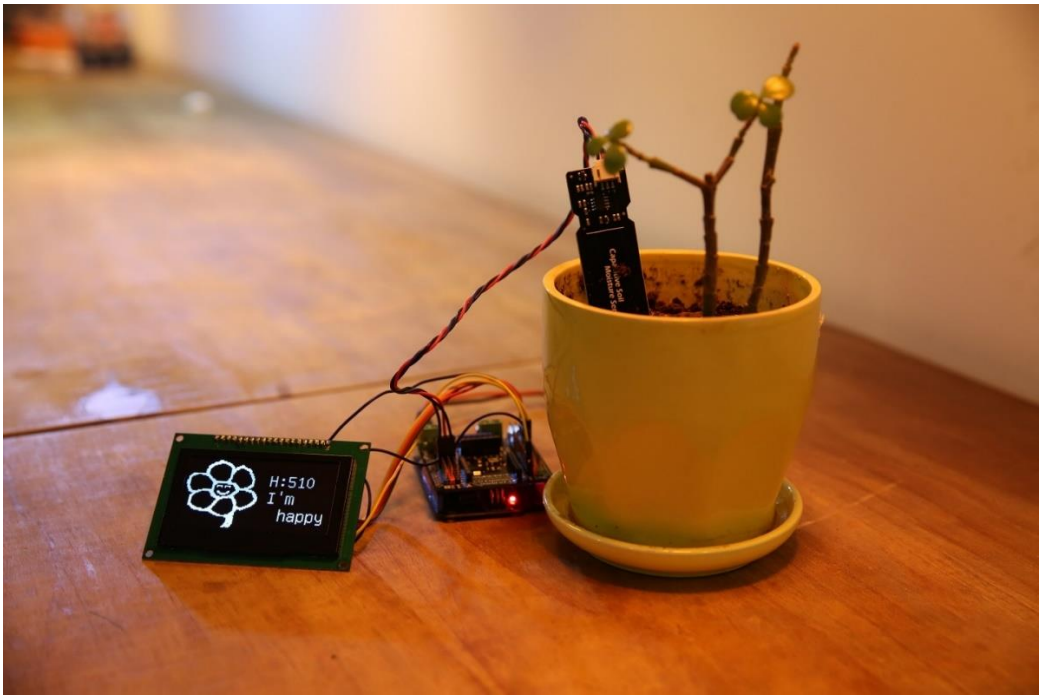

Рисунок 4.7 – «Робота датчику вологості грунту»

4.3 Висновки до розділу 4

1. Існує велике різноманіття датчиків, нижче приведені усі основні категорії датчиків: Датчики звуку та ультразвукові датчики приближення: датчики освітленості та ультрафіолетового випромінювання, датчики вогню, датчики температури та вологості, датчики – акселерометри та гіроскопи, датчики вимірювання сили струму або напруги, датчики газу або диму, датчики дощу, датчики вимірювання РН та для хімічного аналізу
2. До нашої плати можна під’єднати усі ці датчики. Вони можуть бути від одного виробника(Рис.4.1), у нашому випадку це Elecrow, так і від інших виробників.
3. Показано створення найпростішої програми для нових датчиків.

ВИСНОВКИ

Дана робота була присвячена базису інтернету речей, у якому, як було показано, що немає нічого складного, і усе повністю доступне для кожного.

Інтернет речей – це майбутнє, яке все ж нами.

Однак, найголовнішою проблемою на сьогоднішній день є відсутність стандартів в цій галузі, що ускладнює можливість інтеграції пропонованих на ринку рішень і багато в чому стримує появу нових.

Так само для повноцінного функціонування такої мережі необхідна автономність всіх «речей», тобто датчики повинні навчитися отримувати енергію з навколишнього середовища, а не працювати від батарейок, як це відбувається зараз.

Наявність величезної мережі, яка контролює весь навколишній світ, глобальна відкритість даних та інші особливості можуть мати і негативні наслідки. Я думаю, кожен сам може скласти собі список можливих загроз і проблем, які несе в собі ця технологія.

Метою цієї роботи є знайомство з базисом інтернету речей та створення свого власного проекту з вимірювання показників навколишнього середовища та отримання їх на смартфон або комп’ютер через сайт.

У ході дослідження були вирішені такі завдання:

1. Обгрунтувана актуальність розробки власного IoT – рішення
2. Побудовано IoT – рішення на прикладі плати Elecrow ESP8266
3. IoT – рішення було модернізоване до ресурсоефективного

Отже, с першого розділу було проаналізовано, що інтернет речей - це не просто безліч різних приладів і датчиків, об'єднаних між собою дротяними і бездротовими каналами зв'язку і підключених до мережі Інтернет, а це більш тісна інтеграція реального та віртуального світів, в якому спілкування виробляється між людьми і пристроями.

У другому розділі було знайомство з рішенням від компанії Elecrow під назвою ESP8266 IoT Weather Station Kit. Більш детально розглянуті інтерфейси UART та I2C. Зроблен алгоритм програмування.

Проаналізован потенціал цього рішення.

У третьому розділі було досліджено, що IoT-платформа - це рішення, що забезпечують уніфіковану взаємодію між кінцевими пристроями IoT і сервісами обробки даних. Була обрана одна з найбільших платформ – Blynk.

Blynk - хмарний сервіс для створення графічних пультів управління і підходить для широкого спектра мікрокомп'ютерів і мікроконтролерів.

У четвертому розділі було наведено усе різноманіття датчиків, та розкрита їх сумісніть з використованою платою.

Також було показано створення найпростішої програми для для датчиків вогню, звуку та вологості грунту для подальшої інтеграції.

ПЕРЕЛІК ПОСИЛАНЬ

1. «Blynk: простое управление Raspberry и Arduino»: https://lifehacker.ru/blynk/

2. «Common Communication Peripherals on the Arduino: UART, I2C, and SPI»: https://maker.pro/arduino/tutorial/common-communication-peripherals-on-the-arduino-uart-i2c-and-spi

3. «Что нужно знать об интернете вещей: фундаментальный ликбез»: https://habr.com/ru/company/toshibarus/blog/473024/

4. «Інтернет речей»: https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82_%D1%80%D0%B5%D1%87%D0%B5%D0%B9

5. «Що таке інтернет речей і навіщо він потрібен?!» https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82_%D1%80%D0%B5%D1%87%D0%B5%D0%B9

6. «Що таке інтернет речей?»: http://iot.lviv.ua/%D1%89%D0%BE-%D1%82%D0%B0%D0%BA%D0%B5-%D1%96%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82-%D1%80%D0%B5%D1%87%D0%B5%D0%B9/

7. «Що таке інтернет речей?»: https://tokar.ua/read/26780

8. «What is the Internet of Things»: https://www.wired.co.uk/article/internet-of-things-what-is-explained-iot

9. «Internet of things (IoT)»: https://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT

10. «What is the IoT? Everything you need to know about the Internet of Things right now»: https://www.zdnet.com/article/what-is-the-internet-of-things-everything-you-need-to-know-about-the-iot-right-now/

11. «Интернет вещей»: https://www.it.ua/ru/knowledge-base/technology-innovation/internet-veschej-internet-of-things-iot

12. «Интернет вещей: как технологии будущего упрощают настоящее

»

13. «Интернет вещей(IoT)»: https://deps.ua/katalog/iot.html

14. «История Интернета вещей. С чего всё начиналось?»: https://perenio.ua/ru/blog/the-history-of-the-internet-of-things

15. «AWS IoT»: https://aws.amazon.com/ru/iot/?sc_channel=PS&sc_campaign=acquisition_UA&sc_publisher=google&sc_medium=iot_nb&sc_content=iot_e&sc_detail=iot&sc_category=iot&sc_segment=153184138484&sc_matchtype=e&sc_country=UA&s_kwcid=AL!4422!3!153184138484!e!!g!!iot&ef_id=CjwKCAjw8df2BRA3EiwAvfZWaAbPJFogNFJKoVbV7AQr7hS9Evh-vrxSAyG1vQ1GrahqEbQinNKnBRoCwpkQAvD_BwE:G:s&s_kwcid=AL!4422!3!153184138484!e!!g!!iot

ДОДАТОК А

```
#include <ESP8266WiFi.h>
#define MAX_SRV_CLIENTS 3   //Maximum number of connection at the same time, the number of devices you want to access, 8266 tcpserver access only five

const char *ssid = "GorNetwork"; //Here is my wifi, you shold modifi to you when you use wi-fi ssid to connect
const char *password = "23091998"; //Here is my wif password,you should use wifi password of yours

WiFiServer server(8266);//Your port, modify, the range of 0-65535
WiFiClient serverClients[MAX_SRV_CLIENTS];



#define water_PIN 4

#define tem_PIN 5

int n=0;
WiFiClient client;







#define uv_PIN A0


int b,bb,bbb,bbbb,bbbbb,bbbbbb,bbbbbbb;


void setup()
{
   Serial.begin(115200);
```

```
   delay(10);
   pinMode(16, OUTPUT);
   digitalWrite(16, 0);
   WiFi.begin(ssid, password);

   while (WiFi.status() != WL_CONNECTED)
   {
      delay(500);
   }
    Serial.println("WiFi connected");
   Serial.println("IP address: ");
   Serial.println(WiFi.localIP());//WiFi. LocalIP () return 8266 IP addresses

   server.begin();
   server.setNoDelay(true);  //Combined with normal only after some

    pinMode(water_PIN, INPUT);
    //digitalWrite(water_PIN,HIGH);
   // pinMode(tem_PIN, OUTPUT);
    pinMode(uv_PIN, INPUT);



}

void loop()
{
   //blink();



int temp;//   temperature
int humi;//   humidity
int tol;//  Check code
int j;
```

```
unsigned int loopCnt;
int chr[40] = {0};//Create an array of Numbers, used to store 40 bit
unsigned long time1;
  bgn:
  delay(2000);
//The output low level 20 ms (> 18 ms)
//A 40 mu s output level
  pinMode(tem_PIN,OUTPUT);
  digitalWrite(tem_PIN,LOW);
  delay(20);
  digitalWrite(tem_PIN,HIGH);
  delayMicroseconds(40);
  digitalWrite(tem_PIN,LOW);
  pinMode(tem_PIN,INPUT);
  //High level response signal
  loopCnt=10000;
  while(digitalRead(tem_PIN) != HIGH)
  {
    if(loopCnt-- == 0)
    {
//If long time don't return to the high level, the output a hint, start over.
      Serial.println("HIGH");
      goto bgn;
    }
  }
  //Low level response signal
  loopCnt=30000;
  while(digitalRead(tem_PIN) != LOW)
  {
    if(loopCnt-- == 0)
    {
//If long time don't return to the low level, the output a hint, start over.
      Serial.println("LOW");
      goto bgn;
    }
```

```
  }
 //Began to read the bit1-40
   for(int i=0;i<40;i++)
  {
   while(digitalRead(tem_PIN) == LOW)
   {}
 //When there is a high electricity at ordinary times, take time to "time"
   time1 = micros();
   while(digitalRead(tem_PIN) == HIGH)
   {}
 //When there is a low level, write down the time, minus the storage time just now
 //It is concluded that if the value of more than 50 mu s, it is' 1 ', or as' 0 '
 //And stored into an array
   if (micros() - time1  >50)
   {
    chr[i]=1;
   }else{
    chr[i]=0;
   }
  }

 //Humidity, eight bit, converted into numerical values
 humi=chr[0]*128+chr[1]*64+chr[2]*32+chr[3]*16+chr[4]*8+chr[5]*4+chr[6]*2+chr[7];

 //Temperature, 8 bit, converted into numerical values
 temp=chr[16]*128+chr[17]*64+chr[18]*32+chr[19]*16+chr[20]*8+chr[21]*4+chr[22]*
2+chr[23];
  //Check code, 8 bit, converted into numerical values
 //tol=chr[32]*128+chr[33]*64+chr[34]*32+chr[35]*16+chr[36]*8+chr[37]*4+chr[38]*2
+chr[39];
 //Output: temperature, humidity, proofreading yards
 //  Serial.print("temp:");
 //  Serial.println(temp);
  String str1="mode=up&apikey=674c0a5616458361&data={ck001002";
  str1+=temp;
```

```
str1+=".";
str1+=humi;
str1+="}\r\n";
//client.print(str1);
//Serial.print(str1);
Serial.print("temp:");
Serial.println(temp);
bb=temp;
Serial.print("humi:");
Serial.println(humi);
bbb=humi;
```

```
 static long previousMillis = 0;
 static int currstate = 0;

 if (millis() - previousMillis > 200)  //200ms
 {
    previousMillis = millis();
    currstate = 1 - currstate;
    digitalWrite(16, currstate);
     b=analogRead(A0);
     Serial.println(b);

     //Serial.println(bb);
 }

 uint8_t i;
```

```
if (server.hasClient())
{
   for (i = 0; i < MAX_SRV_CLIENTS; i++)
   {
      if (!serverClients[i] || !serverClients[i].connected())
      {
         if (serverClients[i]) serverClients[i].stop();//Not join, release
         serverClients[i] = server.available();//Assign a new
         continue;
      }

   }
   WiFiClient serverClient = server.available();
   serverClient.stop();
}
for (i = 0; i < MAX_SRV_CLIENTS; i++)
{
   if (serverClients[i] && serverClients[i].connected())
   {
      digitalWrite(16, 0);//There is a link, has been long bright

      if (serverClients[i].available())
      {
         while (serverClients[i].available())
            Serial.write(serverClients[i].read());
      }
   }
}
//if (Serial.available())

   //size_t len = Serial.available();
   //uint8_t sbuf[len];
   size_t s=8;
   uint8_t buff[s];
```

```
  int a;
//   a=digitalRead(water_PIN);
   buff[0]='R';
 if(digitalRead(water_PIN)==HIGH)
 {
  buff[1]='1';
  }
  else
  {
    buff[1]='0';
  }



  if(b>450)
  {
    buff[2]='6';
  }
  else
  {
    buff[2]='9';
  }



  bbbb=bbb/10;
  if(bbbb==0)
  {
    buff[3]='A';
  }
  if(bbbb==1)
  {
    buff[3]='B';
  }
  if(bbbb==2)
```

```
{
  buff[3]='C';
}
if(bbbb==3)
{
  buff[3]='D';
}
if(bbbb==4)
{
  buff[3]='E';
}
if(bbbb==5)
{
   buff[3]='F';
}
if(bbbb==6)
{
  buff[3]='G';
}
if(bbbb==7)
{
  buff[3]='H';
}
if(bbbb==8)
{
  buff[3]='I';
}
if(bbbb==9)
{
  buff[3]='J';
}
bbbbb=bbb%10;
if(bbbbb==0)
{
  buff[4]='a';
```

```
}
if(bbbbb==1)
{
  buff[4]='b';
}
if(bbbbb==2)
{
  buff[4]='c';
}
if(bbbbb==3)
{
  buff[4]='d';
}
if(bbbbb==4)
{
  buff[4]='e';
}
if(bbbbb==5)
{
  buff[4]='f';
}
if(bbbbb==6)
{
  buff[4]='g';
}
if(bbbbb==7)
{
  buff[4]='h';
}
if(bbbbb==8)
{
  buff[4]='i';
}
if(bbbbb==9)
{
```

```
  buff[4]='j';
 }

//
// buff[0]=a;
 //buff[1]='1';
 //buff[2]='c';
 //buff[3]='d';
 //Serial.readBytes(sbuf, len);
 bbbbb=bb/10;
 if(bbbbb==0)
 {
  buff[5]='K';
 }
 if(bbbbb==1)
 {
  buff[5]='L';
 }
 if(bbbbb==2)
 {
  buff[5]='M';
 }
 if(bbbbb==3)
 {
  buff[5]='N';
 }
 if(bbbbb==4)
 {
  buff[5]='O';
 }
 if(bbbbb==5)
 {
   buff[5]='P';
 }
 if(bbbbb==6)
```

```
{
  buff[5]='Q';
}
if(bbbbb==7)
{
  buff[5]='U';
}
if(bbbbb==8)
{
  buff[5]='S';
}
if(bbbbb==9)
{
  buff[5]='T';
}

bbbbbb=bb%10;
if(bbbbbb==0)
{
  buff[6]='k';
}
if(bbbbbb==1)
{
  buff[6]='l';
}
if(bbbbbb==2)
{
  buff[6]='m';
}
if(bbbbbb==3)
{
  buff[6]='n';
}
if(bbbbbb==4)
{
```

```
  buff[6]='o';
}
if(bbbbbb==5)
{
   buff[6]='p';
}
if(bbbbbb==6)
{
  buff[6]='q';
}
if(bbbbbb==7)
{
  buff[6]='r';
}
if(bbbbbb==8)
{
  buff[6]='s';
}
if(bbbbbb==9)
{
  buff[6]='t';
}
 buff[7]='Z';



//push UART data to all connected telnet clients
for (i = 0; i < MAX_SRV_CLIENTS; i++)
{
    if (serverClients[i] && serverClients[i].connected())
    {
       // serverClients[i].write(sbuf, len);  //To all the client sends data
       serverClients[i].write(buff,s);
       //delay();
    }
```

```
        }



        void uv();
    }



//void blink()
//{
//    static long previousMillis = 0;
//    static int currstate = 0;
//
//    if (millis() - previousMillis > 200)  //200ms
//    {
//        previousMillis = millis();
//        currstate = 1 - currstate;
//        digitalWrite(16, currstate);
//    }
//}
```

```
void uv()
{
  //int uvLevel = averageAnalogRead(uv_PIN);
  int uvLevel = analogRead(uv_PIN);
  //int refLevel = averageAnalogRead(REF_3V3);
  int refLevel=1023;
  //Use the 3.3V power pin as a reference to get a very accurate output value from sensor
  int outputVoltage = 3.3 / refLevel * uvLevel;
  int v;
  v=outputVoltage;
//  Serial.println(v);
//            size_t u=1;
//        uint8_t buff2[u];
//
//
//
//        buff2[u]=v;
//
//
//        int i;
//        for (i = 0; i < MAX_SRV_CLIENTS; i++)
//        {
//           if (serverClients[i] && serverClients[i].connected())
//           {
//              // serverClients[i].write(sbuf, len);
//              serverClients[i].write(buff2,u);
//              delay(1);
//           }
//        }
}
```

```
//Takes an average of readings on a given pin
//Returns the average
```

ДОДАТОК Б

```
#include <ESP8266WiFi.h>

#include <Adafruit_ssd1306syp.h>
#define SDA_PIN 14//oled_SDA
#define SCL_PIN 12//oled_SCL

#define relay1 5
#define relay  4
const char *ssid     = "GorNetwork";//ere is my wifi, modified to you when you use wi-fi ssid to connect
const char *password = "23091998";//wifi password
const char *host = "192.168.1.110";//TcpServer server's IP address, that is, server IP in the router
WiFiClient client;
const int tcpPort = 8266;//Amended as you build the Server Server port number

  Adafruit_ssd1306syp display(SDA_PIN, SCL_PIN);
  int a,b,i=0,t=0,bb,bbb;

void setup()
{
   Serial.begin(115200);
   delay(10);
   Serial.println();
   Serial.println();
   Serial.print("Connecting to ");//Write a few words of prompt
   Serial.println(ssid);

   WiFi.begin(ssid, password);

   while (WiFi.status() != WL_CONNECTED)//WiFi. The status (), this function is WiFi connection status, return to WiFi link state
```

```
                                    //Not go into it the data returned one, interested to
ESP8266WiFi. The view in the CPP
          {
            delay(500);
            Serial.print(".");
          }//If there is no even lead to a serial port to send...

          Serial.println("");
          Serial.println("WiFi connected");
          Serial.println("IP address: ");
          Serial.println(WiFi.localIP());//WiFi. LocalIP () return 8266 IP addresses


         display.initialize();//Oled initialization

         display.setTextColor(WHITE);//Set the color oled text
         display.setTextSize(1);//Set the oled text size
         display.setCursor(0, 0); //set the oled pointer position
         display.print("Hello,Welcome to Elecrow");//Oled display text
         display.update();
         display.clear();
        }


        void loop()
        {
          while (!client.connected())//A few of connection exception handling
          {
            if (!client.connect(host, tcpPort))
            {
              Serial.println("connection....");
              //client.stop();
              delay(500);

            }
```

```
    }
    while (client.available())//Changes will be here, forwarded to the wireless access to the data to a serial port
    {
      uint8_t c = client.read();
//              int v;
//       v=digitalRead(relay1);
//       Serial.println(v);
       Serial.write(c);
       //Serial.println(c);
      size_t l=8;
      uint8_t buff[l];
//       if(c=='Z')
//       {
//         t=0;
//         i=0;
//         Serial.print(buff[0]);
//         Serial.print(buff[1]);
//         Serial.print(buff[2]);
//         Serial.print(buff[3]);
//         Serial.print(buff[4]);
//         Serial.print(buff[5]);
//        }
//       if(t==1)
//       {
//         buff[i]=c;
//         i++;
//       }
//
//       if(c=='R')
//       {
//         i=0;
//         t=1;
//        }
      //buff[l]=c;
```

```
  // b=buff[1];
//display.clear();
display.setCursor(0, 0);
display.setTextSize(1);
display.print("rian:");//Oled display text
display.setCursor(0, 17);
display.setTextSize(1);
display.print("humi:");//Oled display text
display.setCursor(64, 17);
display.setTextSize(1);
display.print("%");//Oled display text
display.setCursor(0, 34);
display.setTextSize(1);
display.print("temp:");//Oled display text
display.setCursor(64,34);
display.setTextSize(1);
display.print("C");//Oled display text

display.setCursor(0, 51);
display.setTextSize(1);
display.print("uv:");//Oled display text

//display.update();
//delay(10);

     if(c=='1')
 {
   display.setCursor(50,0);
   display.setTextSize(1);
   display.print("no");
  }
 if(c=='0')
 {
    display.setCursor(50,0);
```

```
  display.setTextSize(1);
  display.print("yes");
 }


if(c=='A')
{
  display.setCursor(50,17);
  display.setTextSize(1);
  display.print("0");
}
      if(c=='B')
{
  display.setCursor(50,17);
  display.setTextSize(1);
  display.print("1");
}
      if(c=='C')
{
  display.setCursor(50,17);
  display.setTextSize(1);
  display.print("2");
}
      if(c=='D')
{
  display.setCursor(50,17);
  display.setTextSize(1);
  display.print("3");
}
      if(c=='E')
{
  display.setCursor(50,17);
  display.setTextSize(1);
  display.print("4");
}
```

```
      if(c=='F')
{
  display.setCursor(50,17);
  display.setTextSize(1);
  display.print("5");
}
      if(c=='G')
{
  display.setCursor(50,17);
  display.setTextSize(1);
  display.print("6");
}
      if(c=='H')
{
  display.setCursor(50,17);
  display.setTextSize(1);
  display.print("7");
}
            if(c=='I')
{
  display.setCursor(50,17);
  display.setTextSize(1);
  display.print("8");
}
            if(c=='J')
{
  display.setCursor(50,17);
  display.setTextSize(1);
  display.print("9");
}
      if(c=='a')
{
  display.setCursor(57,17);
  display.setTextSize(1);
  display.print("0");
```

```
}
    if(c=='b')
{
  display.setCursor(57,17);
  display.setTextSize(1);
  display.print("1");
}
    if(c=='c')
{
  display.setCursor(57,17);
  display.setTextSize(1);
  display.print("2");
}
    if(c=='d')
{
  display.setCursor(57,17);
  display.setTextSize(1);
  display.print("3");
}
    if(c=='e')
{
  display.setCursor(57,17);
  display.setTextSize(1);
  display.print("4");
}
    if(c=='f')
{
  display.setCursor(57,17);
  display.setTextSize(1);
  display.print("5");
}
    if(c=='g')
{
  display.setCursor(57,17);
  display.setTextSize(1);
```

```
  display.print("6");
}
      if(c=='h')
{
  display.setCursor(57,17);
  display.setTextSize(1);
  display.print("7");
}
      if(c=='i')
{
  display.setCursor(57,17);
  display.setTextSize(1);
  display.print("8");
}
      if(c=='j')
{
  display.setCursor(57,17);
  display.setTextSize(1);
  display.print("9");
}


if(c=='K')
{
  display.setCursor(50,34);
  display.setTextSize(1);
  display.print("0");
}
      if(c=='L')
{
  display.setCursor(50,34);
  display.setTextSize(1);
  display.print("1");
}
      if(c=='M')
```

```
{
  display.setCursor(50,34);
  display.setTextSize(1);
  display.print("2");
}
    if(c=='N')
{
  display.setCursor(50,34);
  display.setTextSize(1);
  display.print("3");
}
    if(c=='O')
{
  display.setCursor(50,34);
  display.setTextSize(1);
  display.print("4");
}
    if(c=='P')
{
  display.setCursor(50,34);
  display.setTextSize(1);
  display.print("5");
}
    if(c=='Q')
{
  display.setCursor(50,34);
  display.setTextSize(1);
  display.print("6");
}
    if(c=='U')
{
  display.setCursor(50,34);
  display.setTextSize(1);
  display.print("7");
}
```

```
            if(c=='S')
{
  display.setCursor(50,34);
  display.setTextSize(1);
  display.print("8");
}
            if(c=='T')
{
  display.setCursor(50,34);
  display.setTextSize(1);
  display.print("9");
}

if(c=='k')
{
  display.setCursor(57,34);
  display.setTextSize(1);
  display.print("0");
}
      if(c=='l')
{
  display.setCursor(57,34);
  display.setTextSize(1);
  display.print("1");
}
      if(c=='m')
{
  display.setCursor(57,34);
  display.setTextSize(1);
  display.print("2");
}
      if(c=='n')
{
  display.setCursor(57,34);
  display.setTextSize(1);
```

```
  display.print("3");
}
    if(c=='o')
{
  display.setCursor(57,34);
  display.setTextSize(1);
  display.print("4");
}
    if(c=='p')
{
  display.setCursor(57,34);
  display.setTextSize(1);
  display.print("5");
}
    if(c=='q')
{
  display.setCursor(57,34);
  display.setTextSize(1);
  display.print("6");
}
    if(c=='r')
{
  display.setCursor(57,34);
  display.setTextSize(1);
  display.print("7");
}
        if(c=='s')
{
  display.setCursor(57,34);
  display.setTextSize(1);
  display.print("8");
}
        if(c=='t')
{
  display.setCursor(57,34);
```

```
            display.setTextSize(1);
            display.print("9");
          }

              if(c=='6')
         {
           display.setCursor(50,51);
           display.setTextSize(1);
           display.print("strong");
           buff[2]='6';
          }
         if(c=='9')
         {
           display.setCursor(50,51);
           display.setTextSize(1);
           display.print("weak");
           buff[2]='9';
          }
         display.update();
         //delay(100);
          //Serial.println(c);
          if(c=='Z')
          {
            display.clear();
          }

       }


    //    if (Serial.available())//Serial port to read the forwarded to wifi, because a serial port is a sends a cache out to send here
    //    {
    //        size_t counti = Serial.available();
    //        uint8_t sbuf[counti];
    //        Serial.readBytes(sbuf, counti);
```

```
//        //client.write(sbuf, counti);
//
//    }
}
```

ДОДАТОК В

Код Програми:

```
#include <ESP8266WiFi.h>
#include "DHT.h"

#include <WiFiClient.h>
#include <ESP8266HTTPClient.h>
#include <Wire.h>

#define DHTTYPE DHT11
const int DHTPin = 5;
DHT dht(DHTPin, DHTTYPE);
String apiKeyValue = "somekeyhere";

//variables
const int d = 3600000; // x/1000 seconds: interval time for loop
const char* ssid = "ITS-Guest";
const char* password = "";
const char* serverName = "http://its-temperature.radio-signal-
telecom.in.ua/temp_post_data.php";
const int roomId = 506; // room id where sensor is located

static char celsiusTemp[7];
static char fahrenheitTemp[7];
static char humidityTemp[7];

void setup() {
  Serial.begin(115200);
  delay(10);

  dht.begin();

  Serial.println();
  Serial.print("Connecting to ");
```

```
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
   delay(500);
   Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
  delay(500);
  Serial.println(WiFi.localIP());
 }

 void loop() {
  if(WiFi.status()== WL_CONNECTED){

   float t = dht.readTemperature();
   delay(1000);

   HTTPClient http;
   http.begin(serverName);
   http.addHeader("Content-Type", "application/x-www-form-urlencoded");
   String httpRequestData = "api_key=" + apiKeyValue + "&temperature=" + String(t) +
"&roomId=" + String(roomId);
   Serial.println("httpRequestData: " + httpRequestData + "; waiting for " + d/1000 + "
seconds,");

   int httpResponseCode = http.POST(httpRequestData);
   if (httpResponseCode>0) {
    Serial.println("HTTP Response code: " + httpResponseCode);
   }
   else {
    Serial.println("Error code: " + httpResponseCode);
   }
```

```
    // Release resources
    http.end();
  }
  else {
    Serial.println("WiFi Disconnected");
  }
  delay(d);
}
```

Код Хосту:

```
<?php

$servername = "localhost";

$dbname = "radiosig_esp_data";
$username = "radiosig_esp_dba_user";
$password = "trololo";

$api_key_value = "somekeyhere";

$api_key = $temperature = $roomId = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $api_key = test_input($_POST["api_key"]);
    if($api_key == $api_key_value) {
        $temperature = test_input($_POST["temperature"]);
                $roomId = test_input($_POST["roomId"]);

        // Create connection
        $conn = new mysqli($servername, $username, $password, $dbname);
        // Check connection
        if ($conn->connect_error) {
            die("Connection failed: " . $conn->connect_error);
        }
```

```
        $sql = "INSERT INTO temperature_values_XX (temperature, roomId)
        VALUES ('" . $temperature . "', '" . $roomId . "')";

        if ($conn->query($sql) === TRUE) {
            echo "New record created successfully";
        }
        else {
            echo "Error: " . $sql . "<br>" . $conn->error;
        }

        $conn->close();
    }
    else {
        echo "Wrong API Key provided";
    }

}
else {
    echo "No data posted with HTTP POST";
}

function test_input($data) {
    $data = trim($data);
    $data = stripslashes($data);
    $data = htmlspecialchars($data);
    return $data;
}
```

Код Сайту:

```
<?php

$servername = "localhost";

$dbname = "radiosig_esp_data";
$username = "radiosig_esp_dba_user";
```

```
$password = "trololo";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
   die("Connection failed: " . $conn->connect_error);
}


//420
$sql_420 = "SELECT id, temperature, reading_time FROM temperature_values_XX
where roomId=420 order by reading_time desc limit 40";
$result_420 = $conn->query($sql_420);
while ($data_420 = $result_420->fetch_assoc()){
   $sensor_data_420[] = $data_420;
}
$readings_time_420 = array_column($sensor_data_420, 'reading_time');
$temperature_420 = json_encode(array_reverse(array_column($sensor_data_420,
'temperature')), JSON_NUMERIC_CHECK);
$reading_time_420 = json_encode(array_reverse($readings_time_420),
JSON_NUMERIC_CHECK);
$result_420->free();


//404
$sql_404 = "SELECT id, temperature, reading_time FROM temperature_values_XX
where roomId=404 order by reading_time desc limit 40";
$result_404 = $conn->query($sql_404);
while ($data_404 = $result_404->fetch_assoc()){
   $sensor_data_404[] = $data_404;
}
$readings_time_404 = array_column($sensor_data_404, 'reading_time');
$temperature_404 = json_encode(array_reverse(array_column($sensor_data_404,
'temperature')), JSON_NUMERIC_CHECK);
```

```
        $reading_time_404 = json_encode(array_reverse($readings_time_404),
JSON_NUMERIC_CHECK);
        $result_404->free();


        //403
        $sql_403 = "SELECT id, temperature, reading_time FROM temperature_values_XX
where roomId=403 order by reading_time desc limit 40";
        $result_403 = $conn->query($sql_403);
        while ($data_403 = $result_403->fetch_assoc()){
            $sensor_data_403[] = $data_403;
        }
        $readings_time_403 = array_column($sensor_data_403, 'reading_time');
        $temperature_403 = json_encode(array_reverse(array_column($sensor_data_403,
'temperature')), JSON_NUMERIC_CHECK);
        $reading_time_403 = json_encode(array_reverse($readings_time_403),
JSON_NUMERIC_CHECK);
        $result_403->free();


        //406
        $sql_406 = "SELECT id, temperature, reading_time FROM temperature_values_XX
where roomId=406 order by reading_time desc limit 40";
        $result_406 = $conn->query($sql_406);
        while ($data_406 = $result_406->fetch_assoc()){
            $sensor_data_406[] = $data_406;
        }
        $readings_time_406 = array_column($sensor_data_406, 'reading_time');
        $temperature_406 = json_encode(array_reverse(array_column($sensor_data_406,
'temperature')), JSON_NUMERIC_CHECK);
        $reading_time_406 = json_encode(array_reverse($readings_time_406),
JSON_NUMERIC_CHECK);
        $result_406->free();
```

```
//506
$sql_506 = "SELECT id, temperature, reading_time FROM temperature_values_XX
where roomId=506 order by reading_time desc limit 40";
$result_506 = $conn->query($sql_506);
while ($data_506 = $result_506->fetch_assoc()){
   $sensor_data_506[] = $data_506;
}
$readings_time_506 = array_column($sensor_data_506, 'reading_time');
$temperature_506 = json_encode(array_reverse(array_column($sensor_data_506,
'temperature')), JSON_NUMERIC_CHECK);
$reading_time_506 = json_encode(array_reverse($readings_time_506),
JSON_NUMERIC_CHECK);
$result_506->free();


$conn->close();
?>

<!DOCTYPE html>
<html>
<meta name="viewport" content="width=device-width, initial-scale=1">

 <script src="https://code.highcharts.com/highcharts.js"></script>
 <style>
  body {
   min-width: 310px;
            max-width: 1280px;
            height: 500px;
   margin: 0 auto;
  }
  h2 {
   font-family: Arial;
   font-size: 2.5rem;
   text-align: center;
  }
```

```
  </style>

<body>

<h2>Temperature in ITS</h2>

<div id="chart-temperature-420" class="container"></div>
<script>
var temperature_420 = <?php echo $temperature_420; ?>;
var reading_time_420 = <?php echo $reading_time_420; ?>;
var chartT_420 = new Highcharts.Chart({
  chart:{ renderTo : 'chart-temperature-420' },
  title: { text: '1. Room: 420' },
  series: [{
    showInLegend: false,
    data: temperature_420
  }],
  plotOptions: {
    line: { animation: false,
      dataLabels: { enabled: true }
    },
    series: { color: '#059e8a' }
  },
  xAxis: {
    type: 'datetime',
    categories: reading_time_420
  },
  yAxis: {
    title: { text: 'Temperature, *C' }
  },
  credits: { enabled: false }
});
</script>

<div id="chart-temperature-404" class="container"></div>
```

```
<script>
var temperature_404 = <?php echo $temperature_404; ?>;
var reading_time_404 = <?php echo $reading_time_404; ?>;
var chartT_404 = new Highcharts.Chart({
  chart:{ renderTo : 'chart-temperature-404' },
  title: { text: '2. Room: 404' },
  series: [{
    showInLegend: false,
    data: temperature_404
  }],
  plotOptions: {
    line: { animation: false,
      dataLabels: { enabled: true }
    },
    series: { color: '#059e8a' }
  },
  xAxis: {
    type: 'datetime',
    categories: reading_time_404
  },
  yAxis: {
    title: { text: 'Temperature, *C' }
  },
  credits: { enabled: false }
});
</script>


<div id="chart-temperature-403" class="container"></div>
<script>
var temperature_403 = <?php echo $temperature_403; ?>;
var reading_time_403 = <?php echo $reading_time_403; ?>;
var chartT_403 = new Highcharts.Chart({
  chart:{ renderTo : 'chart-temperature-403' },
  title: { text: '3. Room: 403' },
```

```
  series: [{
    showInLegend: false,
    data: temperature_403
  }],
  plotOptions: {
    line: { animation: false,
      dataLabels: { enabled: true }
    },
    series: { color: '#059e8a' }
  },
  xAxis: {
    type: 'datetime',
    categories: reading_time_403
  },
  yAxis: {
    title: { text: 'Temperature, *C' }
  },
  credits: { enabled: false }
});
</script>


<div id="chart-temperature-406" class="container"></div>
<script>
var temperature_406 = <?php echo $temperature_406; ?>;
var reading_time_406 = <?php echo $reading_time_406; ?>;
var chartT_406 = new Highcharts.Chart({
  chart:{ renderTo : 'chart-temperature-406' },
  title: { text: '4. Room: 406' },
  series: [{
    showInLegend: false,
    data: temperature_406
  }],
  plotOptions: {
    line: { animation: false,
```

```
    dataLabels: { enabled: true }
   },
   series: { color: '#059e8a' }
  },
  xAxis: {
   type: 'datetime',
   categories: reading_time_406
  },
  yAxis: {
   title: { text: 'Temperature, *C' }
  },
  credits: { enabled: false }
});
</script>


<div id="chart-temperature-506" class="container"></div>
<script>
var temperature_506 = <?php echo $temperature_506; ?>;
var reading_time_506 = <?php echo $reading_time_506; ?>;
var chartT_506 = new Highcharts.Chart({
  chart:{ renderTo : 'chart-temperature-506' },
  title: { text: '5. Room: 506' },
  series: [{
   showInLegend: false,
   data: temperature_506
  }],
  plotOptions: {
   line: { animation: false,
    dataLabels: { enabled: true }
   },
   series: { color: '#059e8a' }
  },
  xAxis: {
   type: 'datetime',
```

```
    categories: reading_time_506
  },
  yAxis: {
    title: { text: 'Temperature, *C' }
  },
  credits: { enabled: false }
});
</script>



</body>
</html>
```